UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

TRABAJO FIN DE MÁSTER

Clasificación de tráfico de red mediante redes neuronales convolucionales

Máster Universitario en Ingeniería de Telecomunicación

Autor: de Diego de Somonte Cañestro, Ignacio

Tutor: López de Vergara Méndez, Jorge E.

Departamento de Tecnología Electrónica y de las Comunicaciones

FECHA: Junio, 2020

Título: Clasificación de tráfico de red mediante redes neuronales convolucionales

Autor: Ignacio de Diego de Somonte Cañestro

Director: Jorge E. López de Vergara Méndez

High Performance Computing and Networking Research Group

Departamento de Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Madrid, junio 2020

Resumen

El objetivo de este Trabajo de Fin de Máster es la implementación de un sistema de redes neuronales convolucionales que se encargue de clasificar el tráfico de red mediante imágenes en 1D, 2D y 3D. Se realiza un estudio del estado del arte de los diferentes métodos que hay en la actualidad de clasificación de tráfico de red y los problemas que hay para la clasificación. Con ello, se ve la necesidad real de desarrollar un nuevo método de clasificación que soluciones los problemas existentes ya que es muy importante a la hora de realizar tareas como la identificación del *malware* o la planificación de una red.

Por ello, se parte de un conjunto de datos experimentales públicos etiquetados que corresponde a diferentes aplicaciones de Internet. A este conjunto de datos se le realiza una serie de filtros para su posterior conversión en imágenes. Una vez que se tienen todas las imágenes de las aplicaciones se pasa a su tratamiento por los modelos de redes neuronales convolucionales.

Tras obtener los resultados de los modelos de redes neuronales, con el objetivo de intentar mejorarlos, se desarrolla un método nuevo método de votación. Este método de votación se utiliza una vez finalizado el tratamiento con la red neuronal y en el que por medio de un diccionario de flujos y distintos pesos en las votaciones se consigue mejorar aún más los resultados.

Por último, se realiza un estudio en el que se compara tanto los resultados obtenidos en un mismo modelo por los diferentes tipos de imágenes como los mejores resultados que se han obtenido de los modelos diseñados. Tras este estudio, se realiza varias propuestas para posibles trabajos futuros.

Palabras clave: clasificación de tráfico, redes neuronales convolucionales, redes neuronales recurrentes, tráfico cifrado, identificación de aplicaciones, aprendizaje profundo, redes LSTM, clasificación por votación

Abstract

The objective of this Master's Thesis is the implementation of a convolutional neural network system that classifies network traffic using 1D, 2D and 3D images. A study is made of the state of the art of the different methods currently available for classifying network traffic and the problems that exist for the classification. With this, there is a real need to develop a new classification method that solves existing problems as it is very important when performing tasks such as malware identification or network planning.

Therefore, it is based on a set of labeled public experimental data that corresponds to different Internet applications. A series of filters are applied to this data set for subsequent conversion into images. Once all the images from the applications are available, they treated by convolutional neural network models.

After obtaining the results of the neural network models, with the aim of trying to improve them, an own voting method is developed. This voting method is used once the treatment with the neural network is finished and by means of a dictionary of flows and different weights in the votes the results are further improved.

Finally, a study is carried out comparing the results obtained in the same model by the different types of images and the best results obtained from the models designed. After this study, several proposals for possible future work are made.

Keywords: traffic classification, convolutional neuronal network, recurrent neuronal network machine learning, encrypted traffic, application identification, deep learning, LSTM network, voting classification.

Agradecimientos

En primer lugar, me gustaría darle las gracias mi tutor Jorge por haberme dado la oportunidad de realizar este Trabajo de Fin de Máster y todas las veces que me ha ayudado a lo largo de este.

También me gustaría agradecer a mis padres y hermanos, por haber estado a mi lado todos estos años y haber confiado en mi en todo momento.

Además, me gustaría darle las gracias a Jorge, amigo y compañero de clase, que me ha acompañado en este camino desde el primer día que entre por la EPS hasta la entrega de este documento. También a mis amigos que me han ayudado y soportado en los momentos de estrés y agobio convirtiéndolos en tranquilidad.

A mi familia

Índice general

Índice de figuras

Índice de tablas

Índice de gráficas

Capítulo 1 Introducción

1.1 Motivación

Hoy en día, se utilizan diversas técnicas para clasificar el tráfico de red, tales como número de puertos, inspección profunda de paquetes, minería de datos o heurística. Todas estas técnicas se han utilizado para una gran cantidad de aplicaciones, como puede ser la identificación del malware, o la asignación a distintas clases de servicios, para decidir qué paquetes pertenecen a una clase de servicios u otra.

Actualmente, una gran cantidad de las técnicas existentes hasta el momento se están quedando obsoletas. Esto se debe a que un porcentaje amplio del tráfico que circula por Internet se encuentra con algún tipo de cifrado y utilizan puertos que estén abiertos (como el 80 y el 443), por lo que no se puede utilizar la clasificación por número de puerto ni la inspección profunda del contenido de los paquetes.

Encontrar nuevas técnicas que permitan clasificar este tipo de tráfico se ha convertido en una tarea compleja. Algunos estudios en los últimos años han planteado nuevas metodologías utilizando redes neuronales convolucionales (CNN, *Convolutional Neural Networks*) [PanWang19] [WeiWang17] para intentar solventar este problema, pudiendo así clasificar distintos tipos de tráfico. Una red neuronal convolucional es un tipo de red neuronal artificial especialmente efectiva para temas de imagen, como la clasificación y segmentación de imágenes entre otras aplicaciones. La búsqueda de nuevas técnicas, utilizando redes convolucionales, se encuentra en pleno auge, debido a los resultados tan prometedores que se están obteniendo.

Por todo lo anterior, en este Trabajo Fin de Máster se plantean aplicar las técnicas mencionadas, con el objetivo de probar si efectivamente son útiles para identificar y clasificar los distintos tipos de tráfico.

Para ello, se partirá de conjuntos de datos públicos que contienen paquetes cifrados etiquetados dependiendo de la aplicación a la que pertenezca, [Brunswick] [ISCX16]. Estos paquetes serán tratados con diferentes filtros dependiendo de la petición del usuario, como pueden ser: paquete completo, tamaño, quitando cabeceras de nivel de enlace, red y transporte. Una vez tratados estos paquetes y habiendo realizado los filtrados deseados, se convertirán en imágenes, ya sea 1D, 2D o 3D, [JinlinWang17], para su posterior tratamiento en las redes neuronales convolucionales. Tras la obtención de las imágenes, se harán dos subconjuntos de paquetes: el primero nos servirá para entrenar la red neuronal y el segundo

para evaluación. En los dos conjuntos se estudiarán medidas de rendimiento, en términos de precisión, sensibilidad, exhaustividad, etc. Por último, se aplica también otras estrategias, un tipo de red convolucional recurrente, LSTM, *Long Short-Term Memory,* [Malhotra15], y un método propuesto de votación sobre paquetes de un mismo flujo, bastante novedoso en estos ámbitos.

1.2 Objetivos del Trabajo Fin de Máster

El principal objetivo que busca perseguir este Trabajo de Fin de Máster es implementar un modelo basado en redes neuronales convolucionales para poder clasificar el tráfico de red, sea o no cifrado, tratando los paquetes como si fueran imágenes. Posteriormente, se diseñará y se incorporará en el modelo un sistema de votación con el que se pretende conseguir mejores resultados.

Para poder cumplir este objetivo global, se han marcado una serie de objetivos parciales. Concretando los objetivos parciales son:

- Estudiar el funcionamiento de las redes neuronales convolucionales. Conocer el funcionamiento de las redes neuronales convolucionales nos hará mejorar los resultados a la hora de crear nuestros propios modelos.

- Aplicar las redes neuronales convolucionales a la clasificación de red. Tras haber estudiado el funcionamiento de las redes neuronales convolucionales y saber cómo funcionan, es el momento de aplicar estos conocimientos realizando los modelos que se utilizaran en este trabajo.

- Comparar los resultados obtenidos con otros métodos de clasificación. Una vez que se hayan implementado todos los modelos y se hayan obtenido los resultados, se procederá a comparar tanto la evolución de los mecanismos desarrollados como los resultados cuando se aplican diferentes técnicas.

Todos estos objetivos parciales nos ayudarán a cumplir el objetivo general de clasificar el tráfico de red, incluso el cifrado, mediante redes neuronales.

1.3 Fases de realización

Durante la realización de este Trabajo de Fin de Máster se ha seguido una estructura muy marcada, con el objetivo de conseguir los objetivos planteados inicialmente:

- Fase 1 – Investigación: Esta primera fase es donde se estudia el estado del arte de aplicaciones de redes neuronales convolucionales y diferentes mecanismos para la clasificación de tráfico de red. También, de diferentes estudios donde ya utilizan las redes neuronales para la clasificación del tráfico de red.

- Fase 2 – Análisis: En esta fase se analiza el problema de clasificación del tráfico de red. El objetivo de esta fase en conseguir un modelo que sea válido para conseguir solucionar el problema plantado.

- Fase 3 – Desarrollo: Esta fase es en la que más tiempo se ha invertido. En aquí donde se realiza todo el desarrollo *software* para poder aplicar los modelos que se han implementado. No solo el desarrollo de los modelos de redes neuronales, sino que también todo desarrollo para adaptar y filtrar el tráfico de red y su posterior conversión en imágenes.

- Fase 4 – Validación resultados: En esta fase, tras haber realizado el trabajo de investigación, análisis y desarrollo de los modelos, es hora de ver los resultados obtenidos y si realmente compiten con otros métodos ya existentes en la actualidad. Si estos resultados no son los deseados se retrocederá a la Fase 3, para intentar mejorarlos. Este proceso se repetirá hasta conseguir una tasa elevada de acierto o se estabilice sin poder mejorarlo.

- Fase 5 – Redacción de la memoria: En esta última fase se realiza toda la redacción, plasmando en el documento las fases anteriores.

Este proyecto ha durado un total de 11 meses y se ha dividido según las etapas anteriores. En la Figura 1-1 se puede observar el diagrama de Gantt que ha seguido el proyecto.

N° Actividad	Horas	ago-19	sep-19	oct-19	nov-19	dic-19	ene-20	feb-20	mar-20	abr-20	may-20	jun-20
T1. Investigación y Análisis	**50**											
T1.1 Identificación de algoritmos encargados de reconocer paquetes												
T1.2 Documentación y estudio del arte												
T2. Desarrollo	**155**											
T2.1 Desarrollo *software* filtrado												
T2.2 Desarrollo *software* conversón imágenes												
T2.3 Desarrollo *software* red neuronal convolucional												
T2.4 Desarrollo *software* red neuronal recurrente												
T2.5 Desarrollo sistema de votación												
T3. Validación Resultados	**65**											
T3.1 Estudio medidas de rendimiento												
T3.2 Comparación de resultados												
T3.3 Extracción de conclusiones												
T4. Redacción de la memoria y defensa del TFM	**30**											

Figura 1-1 Diagrama de Gantt

1.4 Organización de la memoria

La memoria estará organizada mediante los siguientes capítulos:

- Capítulo 2: En este capítulo se analiza el estado del arte, tanto el relativo a los métodos tradicionales de clasificación de tráfico de red, como el de diferentes tipos de redes neuronales y, por último, el estado del arte de diferentes formas de uso de las redes neuronales convolucionales en clasificación de tráfico de red.
- Capítulo 3: Se describen las herramientas utilizadas durante todo el desarrollo *software*.
- Capítulo 4: Se plantean los problemas de la clasificación del tráfico de red que se quiere tratar.
- Capítulo 5: En este capítulo se expone todo el diseño y desarrollo de los modelos propuestos, así como un nuevo método de votación.
- Capítulo 6: Se utilizan los métodos propuestos para la clasificación de tráfico de red mediante redes neuronales convolucionales.
- Capítulo 7: Por último, a modo resumen, se exponen los resultados y conclusiones a las que se ha llegado en los capítulos anteriores y se propondrá diferentes propuestas para trabajos futuros.

Capítulo 2 Estado del arte

2.1 Introducción

Para la realización de este proyecto se han estudiado en profundidad tres aspectos fundamentas: la clasificación del tráfico de red, las redes neuronales y la utilización de las redes neuronales para la clasificación del tráfico de red. Por ello, el estado del arte está dividido en estos tres aspectos. Es muy importante realizar un buen estado del arte para así conocer las técnicas que se han utilizado y las que podrían servir para la clasificación del tráfico de red.

2.2 Clasificación del tráfico de red

El volumen de tráfico que circula a través de las redes de comunicación ha crecido exponencialmente a lo largo de los años. El aumento constante de servicios que proporcionan las redes y con ellos las aplicaciones que hacen uso de ellas, ha hecho esto posible. Desde el principio de la aparición de Internet la gestión del tráfico ha sido un problema, que se ha intentado solucionar con la clasificación del tráfico de red.

La clasificación de tráfico consiste en dividir el tráfico de red en diferentes categorías, donde cada una de estas categorías requiere un tratamiento diferente.

A lo largo de la historia se han utilizado diversos mecanismos para la clasificación del tráfico de red: técnicas de clasificación mediante el uso de puertos, técnicas heurísticas, inspección profunda de datos y minería de datos. Todas estas técnicas han sido muy útiles para solucionar este problema.

En este apartado se describen algunas de las diferentes tecnologías más importantes que se han empleado hasta la actualidad para la clasificación del tráfico de red.

2.2.1 Número de puertos

Es una técnica que para clasificar el tráfico de red se basa en utilizar el número de puerto de las distintas aplicaciones de internet. Hasta hace unos años, la mayoría de las aplicaciones utilizaban número de puertos fijos, por lo que es fácilmente reconocibles a la hora de clasificar.

En la Figura 2-1 se puede ver distintas aplicaciones y su correspondiente número de puerto. Esto era un problema ya que, al ser conocidos los datos del número de puerto, era bastante sencillo utilizar los datos en su contra. Estando en la era de la privacidad, donde se busca la máxima posible, también suponía un gran problema ya que se conocía que aplicación era.

La técnica de número de puertos era muy utilizada en los árboles de decisión, ya que clasificaba a gran parte de las aplicaciones que circulaban por internet.

Número de puerto	Protocolo	Aplicación	Acrónimo
20	TCP	Protocolo de transferencia de archivos (datos)	FTP
21	TCP	Protocolo de transferencia de archivos (control)	FTP
22	TCP	Shell Seguro	SSH
23	TCP	Telnet	–
25	TCP	otocolo simple de transferencia de correo (Simple Mail Transfer Protocol)	SMTP
53	UDP, TCP	Servicio de nombres de dominios	DNS
67	UDP	Protocolo de configuración dinámica de host (servidor)	DHCP
68	UDP	Protocolo de configuración dinámica de host (cliente)	DHCP
69	UDP	Protocolo de transferencia de archivos trivial	TFTP
80	TCP	Protocolo de transferencia de hipertexto	HTTP
110	TCP	Protocolo de oficina de correos versión 3 (Post Office Protocol version 3)	POP3

Figura 2-1 Ejemplo de número de puerto de aplicaciones [Puerto]

Hoy en día este método no resulta efectivo en todos los casos. Muchas de las aplicaciones han migrado y utilizan número de puertos que no sean fijos, consiguiendo así que no puedan ser detectadas por este método.

2.2.2 Heurística

Las técnicas heurísticas son un tipo de técnicas de un carácter más informal y personal, basadas en la experiencia y conocimiento propios de los investigadores. Se pretende solucionar los problemas con los conocimientos que el investigador tiene sobre el tráfico de red en general o de conocimientos determinados de la red a analizar. Algunas técnicas heurísticas son los árboles de decisión, variedad de información de puntos finales, métodos estadísticos…

2.2.3 Inspección profunda de paquetes

La inspección profunda de paquetes, también conocida como DPI (*Deep Packet Inspection*), es una técnica utilizada para el análisis en profundidad de los paquetes que circulan por las redes de comunicación. Los datos de las aplicaciones se transmiten mediante paquetes. Estos paquetes están estructurados y el análisis interno de cada uno de ellos es conocido como inspección profunda de paquetes, y es una técnica muy utilizada por proveedores de Internet, compañías de medios y empresas tecnológicas.

La utilización de esta técnica ha sido y es muy importante para poder monitorizar el tráfico de las aplicaciones, regular los flujos de tráfico, comprender y analizar los problemas que se

han podido ocasionar en la entrega de paquetes a las aplicaciones de la manera óptima. Con esta técnica es muy fácil tomar decisiones a la hora de administrar una red. Si surge algún problema a la hora de administrarlas o de rendimiento con la imagen tan competa que ofrece está técnica se podrán solucionar.

Para considerarse inspección profunda, la inspección tiene que realizarse en cualquier punto de red que no sea el puesto final, es decir el final. Es muy utilizada en la búsqueda de virus, intrusiones, spam o criterios definidos por los administradores, pero siempre antes de llegar al final. En muchos casos actúa como un cortafuego mejorado, ya que los cortafuegos solo pueden ver el comienzo y el final de los paquetes, pero no tienen la capacidad de bloquear los ataques. La inspección profunda de paquetes nos permite inspeccionar más afondo los paquetes, puede obtener información importante como el contenido del paquete y con ello la aplicación a la que pertenece. También puede capturar todo el tráfico proveniente de un servidor especifico.

La técnica se basa en la carga útil de del paquete. Es decir, toda la información que no se encuentre en las cabeceras por debajo del nivel de aplicación. Busca patrones o secuencias de caracteres que se sepan que tienen las aplicaciones deseadas en tiempo real. Hoy en día no es fiable al cien por cien, debido a que muchas de las aplicaciones transmiten su información de manera cifrada.

Las aplicaciones donde se utilizan más esta tecnología son:

- Red de seguridad.
- Anti-malware.
- Filtrado de URL, localizador de recursos uniforme.
- Protocolos y reconocimiento de aplicaciones.
- Gestión de red,
- Facturación y medición del tráfico.
- Cumplimiento de los derechos de autor.

Las aplicaciones anteriores son solo unos cuantos ejemplos de utilización de la técnica, habiendo muchísimas más aplicaciones.

Esta tecnología, aunque puede ser utilizada para resolver problemas y administrar bien las redes tiene también puntos negativos. Unos de los más importantes es que puede ser utilizada como competencia deshonesta entre las empresas del sector, lo que está siendo muy castigado en la actualidad. En la Figura 2-2 se puede ver de forma esquemática en que consiste la inspección profunda.

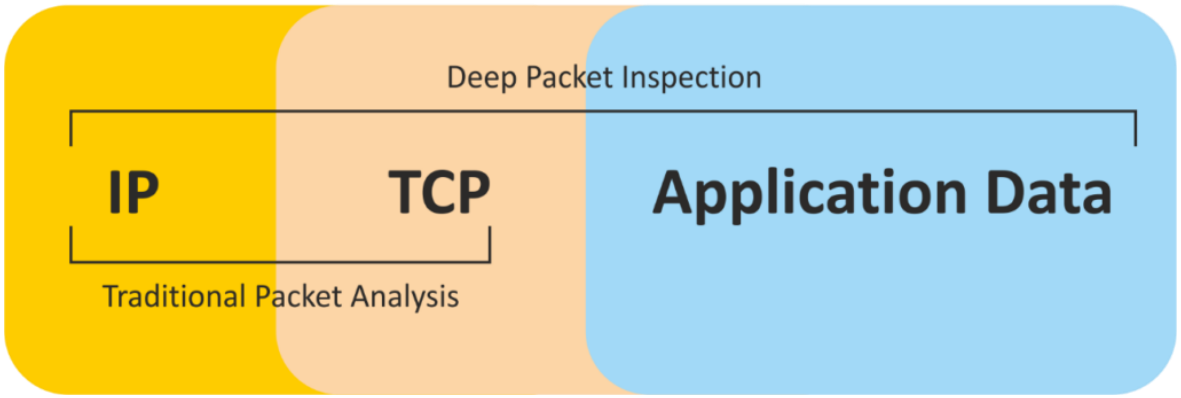


Figura 2-2 Deep Packet Inspection [DPI_im]

2.2.4 Minería de datos

La minería de datos (*Data Mining*) o exploración de datos es una parte de la metodología *Knowledge Discovery in Databases* o KDD [KDD]. Surge por la necesidad de ayudar a comprender una enorme cantidad de datos y poder utilizarlos en beneficio de los usuarios. Por lo tanto, su función principal es la de intentar descubrir patrones repetitivos, tendencias o reglas dentro de grandes conjuntos de datos que permitan solucionar problemas o extraer conclusiones que se puedan utilizar para la toma de decisiones.

Estos patrones se pueden encontrar utilizando técnicas estadísticas o algoritmos de búsqueda como la inteligencia artificial o el aprendizaje automático, ya que gestionan de manera óptima y rápida grandes cantidades de datos.

Los conjuntos de datos son la base para llegar a las conclusiones y transformar estos datos en información que de verdad sean útiles, por lo que es necesario una preparación antes de ser utilizados.

El objetivo principal de la minería de datos es la realización de análisis automáticos o semiautomáticos donde se busca extraer patrones desconocidos, como puede ser la detección de anomalías, reglas de asociación y grupos de registros de datos, más conocido como análisis de clúster. Estos patrones pueden ser vistos como un resumen de la información de entrada más importante, y se pueden utilizar análisis posteriores.

Un proceso de minería de datos común constaría de las siguientes etapas generales:

- Selección del conjunto de datos.
- Análisis de las propiedades de los datos.
- Transformación del conjunto de datos de entrada.
- Selección y aplicación de la técnica de minería de datos.
- Extracción de conocimiento.

- Interpretación y evaluación de datos.

El modelo tendrá que superar todas estas fases, si no fuera así, se podría repetir las fases que se considere oportuno pudiendo ser desde el principio o una fase intermedia. La repetición de las fases se podrá realizar las veces que se considere oportuno hasta conseguir un modelo adecuado y con ello unos resultados óptimos.

Es muy importante la etapa de selección y transformación del conjunto de datos, ya que una mala selección y preparación podría provocar resultado no esperados o erróneos, por lo que el modelo no sería válido

A la hora del diseño es realmente importante tener en cuenta varios factores, en primer lugar, el tipo de datos que se va a utilizar y si es necesario realizar una transformación o preprocesado, por ejemplo, convertir todas las imágenes a un mismo tamaño o cambiar valores de categorías a valores. En segundo lugar. rangos numéricos que puede tener cada dato ya que tienen que estar normalizados pudiéndose utilizar algoritmos de normalización como el *MinMax* o *Logistic* son necesarios en la preparación de los datos a la hora de realizar procesos de minería.

Un proyecto de minería de datos tiene que cumplir mínimo estas cinco fases lineales:

- Comprensión. Alta comprensión el problema que se quiere resolver.
- Determinación. obtención y limpieza. Saber en todo momento de donde proceden los datos necesarios y de donde se van a obtener.
- Creación de modelos matemáticos.
- Validación, comunicación. Comunicar los resultados obtenidos.
- Integración. Integración de los resultados en un sistema transaccional o similar.

Aunque supuestamente debería ser lineal, siempre se puede retroceder de fase para así obtener mejores resultados.

Los análisis de datos mediante la minería de datos pueden aportar numerosas ventajas a las empresas, optimizando su tiempo con una mejorada gestión de los datos y también, para la obtención de clientes, ya que les permitirá aumentar sus ventas

2.3 Aprendizaje automático

El aprendizaje automático, también conocido como *machine learning*, es una de las ramas de la inteligencia artificial (AI). El principal objetico de esta rama consiste en elaborar técnicas de aprendizaje automático para conseguir enseñar a los ordenadores, mejorando la experiencia al utilizar las técnicas. Se cran sistemas que puedan aprender por sí solos. Gracias al aprendizaje automático se consigue hacer automáticas una serie de acciones con el objetico de reducir la necesidad de intervención de los seres humanos. Lo que genera una gran ventaja en situaciones en las que hay que manipular grandes cantidades de datos, siendo el trabajo mucho más efectivo.

La gran peculiaridad de estos sistemas consiste en la enorme capacidad para identificar patrones de gran complejidad, muchos de estos patrones serían casi imposible de detectar por los seres humanos.

Eso sí, la máquina no aprende por sí misma, hay que utilizar algoritmos que previamente han sido diseñados, y será estos algoritmos programados los que aprendan gracias a que se modifican con la entrada de un amplio conjunto de datos en la interfaz creada. De este modo, puede predecir situaciones futuras y con ello realizar acciones de manera automática según condiciones que se han establecido.

Los algoritmos o modelos diseñados se entrenan mediante datos etiquetados y pueden estar formadas por distintas capas. Al principio los modelos diseñados aprenden conceptos sencillos, pero a medida que se entrenan, los modelos irán aprendiendo conceptos cada vez más complejos ya que se toman los datos aprendidos en los niveles anteriores y se aprenden unos nuevos conceptos. En algunos casos, los resultados obtenidos tienen una precisión tan elevada que ni el propio ser humano podría conseguir esos resultados.

Las redes neuronales son sistemas artificiales que pretenden emular el comportamiento de las redes neuronales de los sistemas nerviosos biológicos de los animales. Tal es así, que el conjunto de unidades se les denominará neuronas artificiales, y están conectadas entre sí. Las neuronas artificiales están diseñadas para recibir estímulos en forma de señales de las otras neuronas que están conectadas. Como pasaría en las neuronas biológicas, dependiendo del estímulo la neurona enviará una señal a otra neurona (pudiendo ser a más de una). En los siguientes apartados, se profundizará en las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes, que son dos tipos de redes neuronales profundas.

Hay tres tipos de principales de aprendizaje automático:

1. Aprendizaje supervisado. En este primer tipo de aprendizaje, está basado en la información de entrenamiento. A través, de entrenar el sistema mediante los datos etiquetados que se le da de entrada. A medida que se entrene la red irá aprendiendo parámetros, para que una vez que se introduzcan datos en el sistema sin etiquetas, se pueda clasificar en base a los patrones ya conocidos.

2. Aprendizaje no supervisado. Este tipo de aprendizaje tiene el objetivo la compresión de patrones de información. Es el método de entrenamiento que más se parece al que los humanos realizan, por el modo en que se procesan los dato. O se utilizan etiquetas o valores verdaderos.
3. Aprendizaje por refuerzo. El aprendizaje por refuerzo es una técnica basada en la prueba y error, ayuda de funciones que premian o castigan los sucesos. Las funciones optimizarán el sistema.

2.3.1 Conceptos importantes

En este apartado se explicarán los principales conceptos que son necesarios conocer para entender las redes neuronales y su correcto funcionamiento.

2.3.1.1 Tasa de aprendizaje

La tasa de aprendizaje o ganancia es un parámetro de ajuste que controla el tamaño del paso en las iteraciones cuando se mueve hacia la función de pérdidas. Puede haber varias situaciones, si el valor es muy pequeño, más lento recorrerá la pendiente, por lo que necesitará más tiempo para converger pudiendo quedar en un mínimo local indeseable. En cambio, con una tasa de aprendizaje demasiado alta, aunque necesitaría menos tiempo, podrían aparecer oscilaciones que también son indeseables, por lo que nunca llegaría al punto óptimo.

En la Figura 2-3 se observan cuatro ejemplos de diferentes curvas de aprendizaje dependiendo de la ganancia escogida (tasa de aprendizaje).

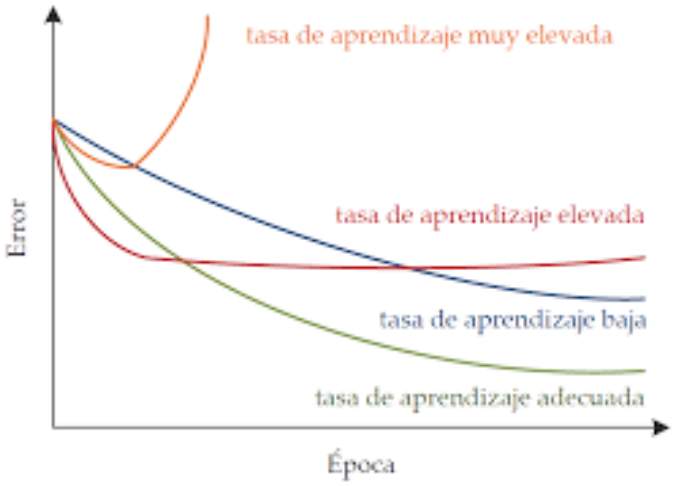


Figura 2-3 Ejemplo tasa aprendizaje [Tasa_aprend]

2.3.1.2 Conjuntos de datos: Entrenamiento/Validación/Prueba

Entrenamiento, validación y prueba son tres conjuntos de datos que se necesitan para entrenar modelos de redes neuronales. A continuación, se explica lo que es cada uno:

- Train. Es el conjunto de datos de entrenamiento. Con este conjunto se realizan todas las pruebas de los modelos. Tiene que ser lo suficientemente grande para que al pasarlo por las redes neuronales tenga resultados significativos a la hora de recoger características. También, debe representar a todo el conjunto de datos, no vale por ejemplo que el conjunto de entrenamiento sean imágenes de pelotas y después se pretenda clasificar pelotas, mesas, sillas, etc. Este conjunto de datos deberá estar dividido por etiquetas que corresponderán a las diferentes clases que tendrá la red neuronal. Las imágenes deberán tener el mismo tamaño y estar normalizadas.
- Validación. El conjunto de validación es utilizado para validar el modelo final del conjunto de entrenamiento.
- Test. Sobre el conjunto de prueba se evalúa el modelo que se ha desarrollado. De este conjunto de datos se reportarán la eficacia del modelo según los resultados obtenido.

En la Figura 2-4 se muestra un ejemplo de estos tres conjuntos. Para este ejemplo, el conjunto de entrenamiento tiene el 50% de los datos, el 25% el conjunto de validación y por último está el conjunto de test con un 25% de los datos. Como se ha explicado el conjunto de entrenamiento suele ser superior al de test para poder hacer un buen entrenamiento del modelo. Los porcentajes de cada uno es elegido por el desarrollador, no hay unos porcentajes fijos, aunque siempre es recomendable que haya una cantidad significativa en cada uno de los conjuntos.

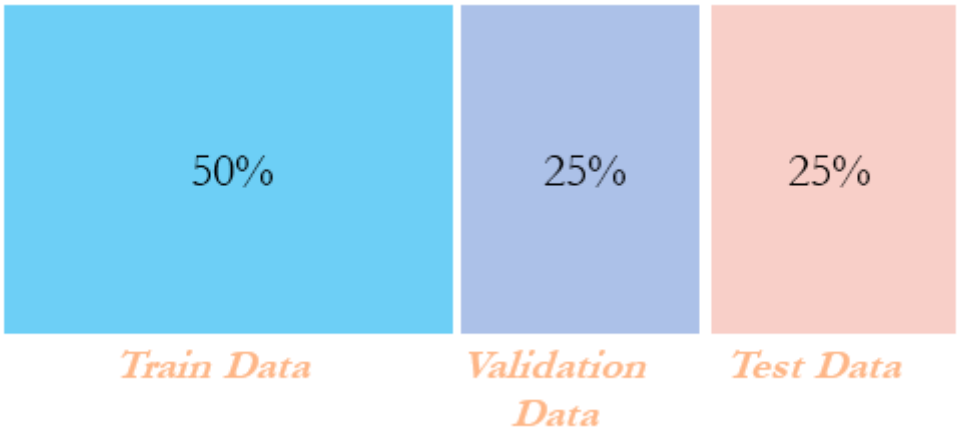


Figura 2-4 Ejemplo Conjunto de datos [TVT]

2.3.1.3 Época e iteración

Se le llama época (*epoch*) a cada iteración de la red neuronal por el conjunto de datos de entrada de entrenamiento en las que haya ajuste de variables. El ajuste de variables se puede hacer después de conocer los vectores de entrada individualmente o por lotes, también se pueden realizan pasadas hacia delante y hacia atrás para actualizar el ajuste.

Si el número de épocas es muy elevado, la veces que se actualizan las variables durante las capas de las redes neuronales será mayor. Este efecto podría provocar sobreajuste o, al contrario, si el número de épocas es muy bajo, se produciría un ajuste insuficiente. Por lo que es muy importe establecer bien el número de épocas que va a realizar la red neuronal.

Una iteración es un solo ajuste de las variables de entrenamiento en una red neuronal. Una red neuronal con una solo iteración corre el riesgo de que no entrene correctamente por lo que no se obtendrán bueno resultados. Como las redes neuronales necesitan aprender de los datos de entradas es necesario definir un número de épocas adecuado.

2.3.1.4 Tamaño del lote

El tamaño del lote o *batch* es un parámetro de las redes neuronales que se utiliza para dividir el conjunto de datos con el que se trabaja. Esto es realmente útil ya que, si se trabajase con todo el conjunto de datos, el coste computacional sería muy costoso y lento. Con conjunto de datos pequeños esto no sería necesario ya que no se notaría.

Para conjunto de datos muy grandes y modelos de redes neuronales complejas lo más habitual es utilizar un tamaño de lote reducido así se consigue que el entrenamiento sea más rápido. Aunque podría ocurrir que con un tamaño más elevado el tiempo fuera menor. Debido a esto, la opción más recomendable sería realizar un análisis profundo para localizar el tamaño adecuado para cada conjunto de datos.

2.3.1.5 Normalización de lotes

Es la capa de las redes neuronales que se encarga de normalizar los conjuntos de datos, tamaño del lote. Es muy importante que los datos introducidos se normalicen ya que si no podrían tener distancias muy dispares entre ellos.

Las imágenes por ejemplo pueden tener valores que comprenden de 0 a 255. Cuando normalizamos los datos, se consigue que las distancias de los datos vayan de 0 a 1. Este proceso es realmente útil ya que ayuda a la red neuronal a trabajar mejor cometiendo menos errores. Pero esta normalización solo se realiza a la primera para la primera capa, conforme que los datos pasan por otras capas la normalización se pierde poco a poco, así que cuando tenemos una red neuronal con muchas capas puede llegar a desaparecer por completo.

Con el método de normalización de lotes se consigue normalizar los datos justo antes de que pasen por la función de activación en cada una de las capas, es decir, tendremos siempre normalizados los datos.

2.3.1.6 Capa de *Dropout*

La capa de *Dropout* es una capa que se encarga de regularizar y es muy utilizada para evitar el sobreentrenamiento, también conocido como *overfitting*, en las redes neuronales.

Esta técnica desactivará de forma aleatoria en el proceso de entrenamiento el porcentaje de neuronas que se le indique. Por cada capa que tenga la red neuronal el porcentaje de *Dropout* puede cambiar, en las primeras capas se suele utilizar un porcentaje muy alto para mantener a la mayoría de las neuronas activadas y en las últimas un porcentaje intermedio. En cada iteración de la red neuronal con la capa *Dropout* se desactivarán neuronas diferentes. Se consigue así que las neuronas no memoricen parte de las entradas, proceso al que se le llama sobreajuste. En la Figura 2-5 se puede visualizar un ejemplo sencillo de una capa *Dropout*, desactivándose una serie de neuronas al azar marcadas en rojo.

El proceso hace que la red neuronal aprenda características sólidas que serán más útiles para el resto de las neuronas de su entorno. Y otra consecuencia de utilizar esta capa, es que las neuronas se vuelven menos dependientes de la salida de las neuronas conectadas.

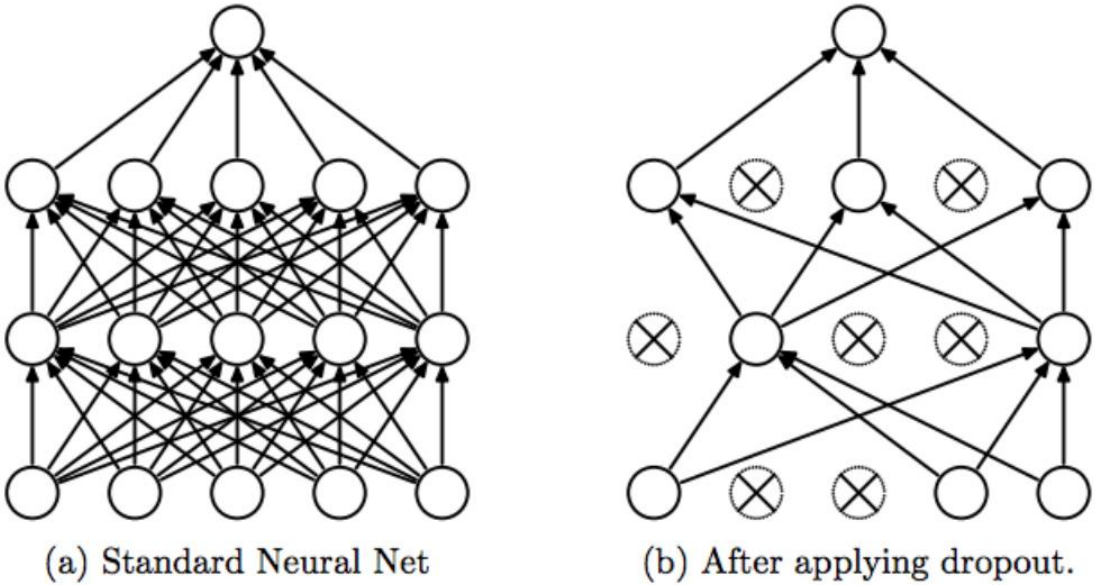


Figura 2-5 Ejemplo *Dropout* [Dropout_im]

2.3.2 Redes neuronales convolucionales

Las redes neuronales convolucionales, CNN, son un tipo de red neuronal artificial que se utilizan para el procesamiento de imágenes. Su mecanismo está basado en operaciones de convolución. Este tipo de red es una variación de un perceptrón multicapa, pero debido a que su aplicación se realiza en matrices bidimensionales, son muy efectivas en tareas de reconocimiento de imágenes.

Estas redes consisten en múltiples capas de filtros convolucionales de diferentes dimensiones. A medida que se va avanzando por las capas, se disminuye la dimensionalidad.

Las neuronas de capas lejanas son mucho menos sensibles a cambios en los datos de entrada. Al utilizar convoluciones se reduce la carga computacional del sistema.

Existen numerosas redes neuronales convolucionales que ofrecen muy buenos resultados a la hora de la clasificación de imágenes. En Figura 2-6 se puede ver unos de los más importantes, la red neuronal AlexNet.

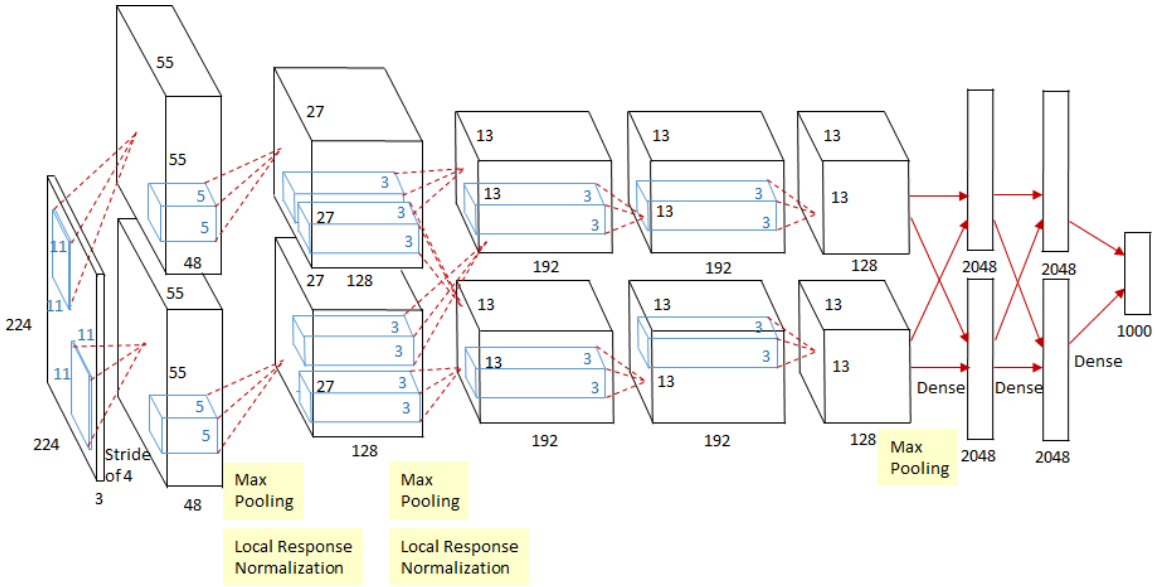


Figura 2-6 Ejemplo de red neuronal convolucional AlexNet [Alexnet_im]

2.3.3 Redes neuronales recurrentes

Las redes neuronales recurrentes, RNN, son un tipo de red neuronal artificial que se caracteriza por integrar bucles de retroalimentación. Los bucles permiten que los datos que están siendo utilizados permanezcan durante las épocas de entrenamiento que se ha establecido.

El proceso consiste en conectar la salida de una capa en la entrada. El entrenamiento de una red neuronal recurrente es diferente al de una red neuronal convolucional, ya que hay que prolongarse para cada paso temporal. Al funcionar así, es necesario tener equipo bastante potente debido que consumen demasiada memoria RAM, memoria de acceso aleatorio. Se puede solucionar utilizando el mismo número de capas como de pasos temporales.

Lo que puede suceder es el problema de desvanecimiento de gradiente, aunque se podría solucionar con la incorporación de capas tipo LSTM. En el apartado 5.4.3 se explicará el motivo de utilizar este tipo de red convolucional.

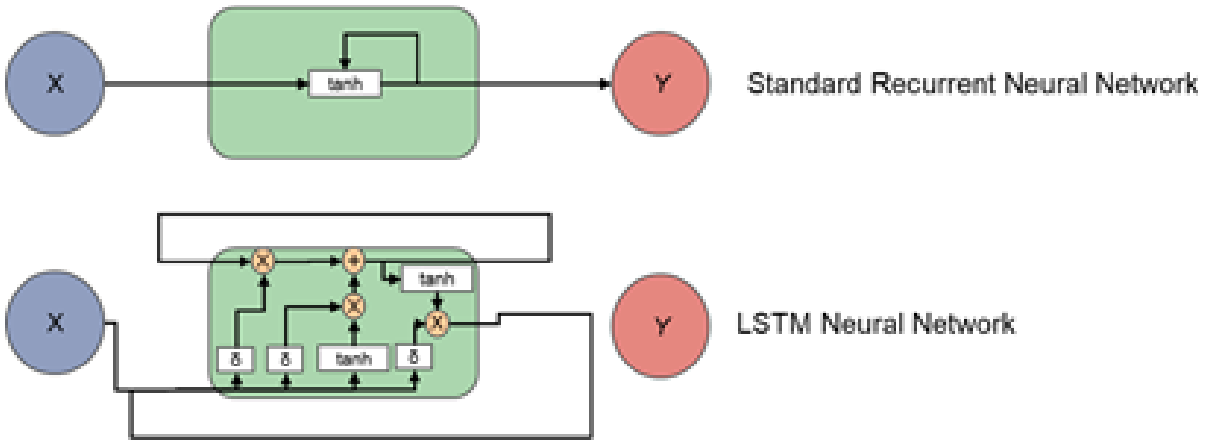


Figura 2-7 Estructura RNN y LSTM [LSTM_im]

La Figura 2-7 muestra las arquitecturas de ambos tipos de redes. Como ya se ha mencionado, la CNN tiene una magnifica capacidad de aprendizaje de las características espaciales, sin embargo, no puede extraer características temporales como los datos de series temporales que también son de gran ayuda para la clasificación. Por lo que se ha propuesto una nueva solución basada en una combinación con CNN y LSTM.

2.4 Clasificación del tráfico con redes neuronales

La utilización de las redes neuronales mediante redes neuronales está en auge actualmente. Se han juntado dos situaciones que ha hecho esto posible, muchas de las técnicas, como se ha ido avanzando en apartados anteriores, no son fiables cien por cien en la actualizada, por lo que ha hecho que los investigadores busquen otras técnicas para poder clasificar el tráfico de red. Una parte de estos científicos se han inclinado a la utilización de redes neuronales, debido a que se sabe que estas redes tienen una gran capacidad para clasificar imágenes, por lo que, si se convirtiera el tráfico de red en imágenes y se realizara un buen modelo con buenas configuraciones, se podría clasificar el tráfico correctamente.

En este apartado se comentarán algunas de las investigaciones que se han tenido en cuenta para realizar este proyecto.

El primer trabajo que se comentará es el llamado "*End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks*" desarrollado por expertos de University os Science and Technology of China y del Institute of Acoustics, Chinese Academy of Sciences [JinlinWang17].

En este estudio se propone un método de clasificación del tráfico cifrado de extremo a extremo con redes neurales convolucionales unidimensional. El método integra la extracción de características, la selección de características y el clasificador en un marco unificado de

extremo a extremo, es decir, desde la entrada de las imágenes sin preprocesar hasta el final con la imagen clasificada. El método se valida con el conjunto de datos públicos de tráfico ISCX VPN-nonVPN (será el que se use en este Trabajo de Fin de Máster) aunque únicamente realizan la clasificación sobre 12 categorías. Obteniendo grandes resultados en los experimentos realizados, habiendo también una comparación con una red neuronal convolucional bidimensional.

El segundo trabajo que se comentará es el llamado "*Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things*" [Martin17] desarrollado por el departamento TSyCeIT, ETSIT, Universidad de Valladolid, y el Instituto de Investigación para la Gestión Integrada de Zonas Costeras, Universitat Politécnica de Valencia.

En este estudio se presenta una nueva técnica para la clasificación de tráfico de red basada en la combinación de modelos de aprendizaje profundo que pueden ser usados para el tráfico. Se combina una red neural convolucional (CNN) con una red neural recurrente (RNN), una vez combinadas se observa que los resultados obtenidos proporcionan los mejores resultados de detección comparado con otros proyectos que utilizan algoritmos alternativos. Realizan un estudio con diferentes arquitecturas RNN, la primera con una capa únicamente y la segunda con dos capas. Clasificando hasta 14 categorías.

Lo más característico de este proyecto es que se demuestra que no es necesario procesar un gran número de paquetes por flujo para tener excelentes resultados: cualquier número de paquetes superior a 5-15 (dependiendo de la arquitectura) da resultados similares.

El tercer trabajo, "*Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning*", desarrollado por Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, Mohammdsadegh Saberian de la Universidad Industrial Sharif, Teherán, Irán. [Lotfollahi12].

En este estudio se propone sistema ten el que se realiza la extracción de las características como la clasificación. Este sistema realiza la caracterización del tráfico en la red, clasificándolo en las principales clases (por ejemplo, FTP, protocolo de transferencia de archivos, y P2P, red de pares), y la identificación de las aplicaciones en las que se desea que el usuario final las identifique (por ejemplo, BitTorrent y Skype). Este sistema se diferencia de otros métodos ya que puede utilizarse para tráfico cifrado. Además, también puede distinguir entre el tráfico de red VPN, red privada virtual, y no VPN. Tras una fase inicial de preprocesamiento de los datos, los paquetes se introducen en un codificador automático apilado que está integrado en el sistema y una vez finalizado por una red neural convolucional para clasificar el tráfico de red. Se utiliza una red neuronal convolucional como modelo de clasificación que consiguió unos resultados del 98% en la tarea de identificación de aplicaciones y 94% en la tarea de categorización del tráfico. Considerándose así uno de los métodos de clasificación más fiables que utiliza el conjunto

de datos de experimentación ISCXVPN-noVPN. Este proyecto clasifica 17 categorías de tráfico de red.

El último trabajo que se va a comentar es el llamado "*Autonomous Model Update Scheme for Deep Learning Based Network Traffic Classifiers*" [JZHANG19], desarrollado por Jielun Zhang, Fuhao Li, Hongyu Wu y Feng Ye pertenecientes a la University of Dayton.

En este estudio, se propone un esquema de actualización de modelo autónomo para filtrar los paquetes de datos de una nueva aplicación del tráfico de red y construir un conjunto de datos de aprendizaje correspondiente; y actualizar el clasificador de tráfico de red actual con aprendizaje de traspaso.

En particular, el núcleo del esquema propuesto es una red neural convolucional que sirve para filtrar y construir un conjunto de datos de nuevos paquetes de aplicaciones a partir del tráfico de red activo. La evaluación se lleva a cabo en base a un conjunto de datos abierto. Los resultados demostraran que el sistema propuesto de actualización del clasificador autónomo puede filtrar con éxito los paquetes de una nueva aplicación del tráfico de la red y construir un conjunto de datos de capacitación correspondiente. Este proyecto es el único de los que se han comentado que realiza pruebas con redes convolucionales de 1D, 2D y 3D.

Como crítica a estos trabajos, se sospecha que han eliminado las categorías que peor clasificaban, ya que ninguno realiza el estudio sobre las 42 categorías que contiene el conjunto de datos experimentales ISCXVPN-noVPN. Al eliminar estas categorías pueden estar consiguiendo tasas de acierto más elevadas, pero no serían verdaderas porque cuando se evaluará otro conjunto de datos que no se supiera a que aplicación pertenece, no sabrían si lo están clasificando correcta o erróneamente.

Hay muchos estudios hoy en día que utilizan las redes neuronales para la clasificación del tráfico de res, estos proyectos han sido solo cuatro ejemplos de ello y en los que más se ha inspirado el Trabajo de Fin de Máster.

2.5 Conclusiones

Durante todas estas fases del estudio del arte se han comentado los métodos y técnicas más importantes tanto de la clasificación del tráfico de red, como las redes neuronales y la utilización de las redes neuronales para la clasificación del tráfico de red.

En un primer lugar se empezó explicado las técnicas de clasificación de tráfico tradicionales, como la clasificación por número de puerto o las técnicas de inspección profunda. Están técnicas era de gran utilidad y tenían grandes ventajas sobre otras técnicas, pero con el transcurso de los años se han ido quedando obsoletas debido a que muchas de las aplicaciones se han ido actualizando para conseguir no ser clasificadas por estas técnicas.

En segundo lugar, se han explicado los principales conceptos de las redes neuronales ya que es muy importante entenderlos para poder realizar la mejor configuración posible a la hora de crear un modelo. También se han explicado las diferencias entre los dos grandes conjuntos

de redes neuronales, las convolucionales y las recurrentes, explicando cuando sería convenientes utilizar cada uno. Aunque ambas son muy adecuadas para la clasificación de cualquier tipo de imágenes.

Por último, se ha podido observar que la utilización de técnicas de aprendiza automático es muy beneficioso. Los clasificadores de última generación basados en el aprendizaje profundo tienen una gran precisión incluso cuando procesan paquetes de datos cifrados. Realizando las etapas de entrenamiento, test y validación con una configuración adecuada, se podría clasificar y conseguir unos resultados bastantes buenos. Por lo que la utilización de las redes neuronales parece que puede ser tanto el presente como el fututo en el ámbito de la clasificación del tráfico de red.

Capítulo 3 Descripción del problema y posibles soluciones

3.1 Introducción

Este capítulo describirá el problema que hay en la actualidad con la clasificación del tráfico de red y las posibles soluciones que se han implementado en este proyecto. Los resultados de las soluciones propuestas, así como el diseño y desarrollo se explicarán en los posteriores capítulos.

3.2 Formulación del problema

En la actualidad existen numerosos métodos para la clasificación del tráfico de red, pero mucho de estos métodos se están quedando obsoletos debido a que utilizan métodos de clasificación por número de puerto o inspección profunda del contenido del paquete. A esto hay que añadirle que hoy en día la mayoría del tráfico que circula por internet perteneciente a cualquier aplicación se encuentra cifrado. Muchas de las aplicaciones a su vez buscan ser lo más privadas posible por lo que intentan que sus datos no sean conocidos. Por lo que buscar un método que resuelva este problema clasificando el tráfico de red con un porcentaje elevado de acierto es una tarea bastante complicada. Es muy importante conocer el tráfico que circula por la red para poder tener una buena gestión y poder preparar la red acorde al tráfico que circula. También es muy importante conocer el tráfico para posibles amenazas ya que se actuaría mucho más rápido y eficientemente, pudiendo conseguir parar el ataque.

En el apartado siguiente se propondrán las tres posibles soluciones con las que se pretende solventar el problema anteriormente descrito.

3.3 Sistemas propuestos

3.3.1 Sistema basado en arquitectura CNN

El primer sistema propuesto para este Trabajo de Fin de Máster es un sistema basado en arquitectura CNN, para ello lo primero que se realizaría es convertir todos los paquetes en imágenes, ya sea 1D, 2D o 3D. Una vez que se tienen todos los paquetes convertidos a imágenes y cada imagen en la carpeta de la aplicación correspondiente se utilizaría un sistema de clasificación de imágenes mediante un modelo CNN.

El reconocimiento de imágenes y su posterior etiquetado automático es una de las aplicaciones más comunes de los modelos CNN, y gracias a estos modelos se han experimentado enormes progresos en este campo. Principalmente están basados en emular los procesos biológicos que sufren las neuronas. Estos procesos biológicos tienen lugar en el córtex visual, donde las neuronas responden individualmente a diferentes estímulos. Por lo que los modelos CNN reconocen ciertas características de las imágenes, ya sea formas parecidas, colores, bordes u otras características, y aprenden de ellas. Esta capacidad es una ventaja única ya que elimina los esfuerzos que se tendría que realizar para reconocer características. Es justamente esta característica la que nos interesa, debido a que estamos trabajando con imágenes creadas a partir del tráfico de red, y es difícil encontrar semejanzas de cualquier tipo. Cuando el ojo humano ve una imagen donde aparece un perro rápidamente lo identifica como tal, pero no ocurre lo mismo cuando se trata de una imagen creada artificialmente y que no representa nada conocido ya que no tiene una estructura donde lo puedas comparar. Un buen modelo CNN tendrá la capacidad de encontrar las características y semejanzas de cada imagen para su posterior clasificación.

Por todo esto, se ha decidido que una arquitectura CNN podría ser muy ventajosa frente a otros métodos de clasificación de tráfico de red, y por eso se desarrollará un modelo que permita clasificar el tráfico de red mediante imágenes.

3.3.2 Sistema basado en arquitectura CNN + LSTM

El segundo sistema propuesto es un sistema basado en arquitectura CNN al que se ha añadido una arquitectura artificial de red neuronal recurrente. Como ocurría en el Sistema basado en arquitectura CNN lo primero que se realiza es convertir todos los paquetes en imágenes para después utilizarlas en u sistema de clasificación de imágenes con un modelo CNN + LSTM. Esta solución se ha propuesto debido a que una red LSTM (*Long Short Term Memory*) es un tipo de RNN (*Red Neuronal Recurrente*) que suele tener muy buenos resultados cuando se trata de aprender de secuencias de datos temporales, lo que es muy atractivo en este proyecto ya que se trabaja con flujo de datos.

Lo que se pretende es que, al trabajar con los flujos del tráfico de red de las aplicaciones, es que se consiga crear una conexión entre un conjunto de paquetes entrantes en el modelo de red neuronal que se implementará. Cuando se está navegando por internet o utilizando cualquier aplicación con conexión a internet, y se realiza una petición no solo se envía un mensaje, sino que se envían varios. Por ejemplo: cuando se está en la aplicación de YouTube y se quiere ver un video, no hay un único mensaje que transmite el video entero, hay un conjunto de flujos por el que se transmite el video. Este conjunto de flujos de paquetes tendrá una estructura muy parecida. Puede ocurrir que en el trascurso de tiempo en el que se envía ese flujo de datos se cuele algún paquete de otra aplicación que se esté consultado en el mismo periodo de tiempo, de aquí la importancia de utilizar también un modelo CNN. Por lo que un modelo CNN al que se incorpora una red LSTM podría ser una buena solución para solventar el problema de clasificación de trafico de red.

3.3.3 Sistema basado en arquitectura CNN + Método votación

La última propuesta que se ha realizado para el Trabajo de Fin de Master es la más novedosa en todo el ámbito de la clasificación de este proyecto. Consiste en dos partes, la primera es la explicada anteriormente en el Sistema basado en arquitectura CNN y la segunda parte es el método de votación que se ha propuesto.

El sistema realizará en primer lugar todo el modelo CNN y una vez terminado se intentará mejorar los resultados obtenidos mediante el método de votación. Este método consiste en tomar una imagen que se ha clasificado con anterioridad y buscar la moda del flujo para identificarlo por votación. La votación se realizará a un cierto número de paquetes del flujo predefinido desde el principio y con distintas ponderaciones, dando más importancia a los primeros paquetes del flujo. Este proceso se realizará con todas las imágenes clasificadas. El método se basa en que el modelo CNN puede equivocarse al clasificar una imagen de un cierto flujo, pero es más difícil que clasifique mal todas las imágenes del mismo flujo, por eso a la hora de hacer la moda del flujo, en la mayoría de los casos podría mejorar los resultados. Este método puede no ser efectivo cuando un flujo de paquetes contiene un único paquete, ya que, al hacer la votación, al haber solamente un único paquete volverá a clasificar igual. En contraposición el método será más efectivo cuando el flujo contenga una cantidad significante de paquetes ya que podrá realizar una votación que realmente sea justa.

Gracias a las características que nos proporcionan los modelos CNN y con este método de votación, se pretende conseguir una tasa de clasificación que pueda competir con los métodos tradicionales de clasificación de trafico de red.

3.4 Conclusiones

Como se ha podido observar se han propuesto tres tipos de soluciones con el principal objetivo de conseguir solucionar de la manera más eficiente el problema de clasificación del tráfico de red. Las tres posibles soluciones que se han propuesto tienen en común que utilizan redes neuronales convolucionales por medio de imágenes, ya que se ha demostrado que las CNN dan fantásticos resultados a la hora de clasificación de imágenes.

El último método es el más novedoso ya que incorpora un método de votación propio que pretende mejorar aún más los resultados obtenidos del sistema basado en arquitectura CNN.

Capítulo 4 Entorno Experimental

En este capítulo se describe el entorno con el que se ha realizado todo el proyecto. Para poder desarrollarlo se han tenido que usar distintas herramientas dependiendo de lo que se quisiese realizar. Cabe destacar que, aunque sean estas las utilizadas, se han tenido en cuenta también otras herramientas, incluso se han llegado a probar para ver cuál resultaba mejor a la hora de realizar la función requerida.

4.1 Conjunto de datos experimentales

El conjunto de datos de experimentación ***ISCXVPN2016*** [ISCX16] de la *University of New Brunswick* [Brunswick], es el conjunto de datos de experimentación utilizado para la evaluación de los sistemas que se propondrán más adelante, son un conjunto de datos públicos disponible únicamente para los investigadores [Drapper16], que contienen paquetes cifrados y etiquetados dependiendo de la aplicación a la que pertenezca.

El contenido de este conjunto de paquetes pertenece a las siguientes aplicaciones que se ha divido dependiendo la función que desempeñe:

- Chat: AIM, Facebook and Hangouts, ICQ y Skype,
- Email: IMAPS (Protocolo de acceso a mensajes de internet), POP3S y SMPTS (Protocolo para transferencia simple de correo).
- Navegadores web: Chrome y Firefox.
- P2P: Bittorrent y uTorrent.
- Streaming: Vimeo y Youtube.
- Transferencia de archivos: FTPS, SFTP (Protocolo de transferencia de archivos SSH) y Skype.
- VoIP: Facebook, Skype and Hangouts voice calls.

Todas estas categorías hacen un total de 42 aplicaciones diferentes dependiendo, como se ha comentado anteriormente, del tipo de aplicación y si el tráfico está cifrado o no. La elección de este conjunto de datos de experimentación es debido a que el algoritmo que se implementa necesita un conjunto de datos de experimentación de imágenes ya clasificadas y etiquetadas.

4.2 Herramientas

4.2.1 Ubuntu Linux

Ubuntu es un sistema operativo con código abierto y *software* libre. Pertenece a una distribución de Linux apoyada en el sistema operativo Debian. La mayoría del *software* disponible para este sistema operativo es de licencia libre y de código abierto, por lo que es muy atractivo para muchos de los usuarios.

Este sistema operativo cuenta con actualizaciones periódicas para no quedarse obsoleto, llegando al punto de que cada 6 meses se publica una versión diferente. Las actualizaciones no son siempre visualmente, sino que pueden ser actualizaciones de seguridad, actualizaciones de programas, incorporación de nuevas funcionalidades, parches para diferentes bugs que pueden ser críticos para el funcionamiento correcto del sistema operativo.

Cabe destacar que una de sus características más notorias es el enorme entorno de programación que facilita y que permite trabajar con todo tipo de lenguaje, por lo que se le saca gran provecho a este sistema operativo. La herramienta que se ha utilizado para este proyecto es *GNU Compiler Collection* (Colección de compiladores GNU), también conocido como GCC.

GNU Compiler Collection es un grupo de compiladores creados por el proyecto GNU y es de *software* libre. Aunque en un principio solo permitía compilar para el lenguaje de programación C, actualmente se ha extendido a otros lenguajes como C++, Ada, Fortran Objetive C, Objetive C++ y otros. El objetivo primordial era mejorar el compilador ya existente en los sistemas GNU.

4.2.2 Wireshark

Wireshark [Wire] es un *software* de libre distribución (*open-source*) cuyo principal objetivo es analizar protocolos de red en tiempo real y fue diseñado por Gerald Combs. Está disponible para distintos sistemas operativos, desde los más importantes como Windows y Linux a otros más pequeños como OpenBSD o NetBSD.

Las redes actuales cuentan con un gran volumen de tráfico y con esta herramienta pueden ser filtrados. Esta amplia gama de filtros, capturan solo el tipo de tráfico deseado para su visualización e inspección. También cuenta con herramientas de búsqueda que hace más rápido y más intuitivo encontrar lo que realmente se desee.

Cuenta con una interfaz sencilla e intuitiva lo que hace más sencillo para el usuario extraer cada uno de los paquetes capturados. Al tener tan cantidad de funciones, proporciona al usuario grandes posibilidades para realizar tareas de análisis de tráfico. Además, nos permite analizar datos capturados en disco.

Existen otras herramientas que tienen funcionalidades parecidas como puedes ser Microsoft Message Analyze anteriormente conocido como Microsoft Network Monitor, Windump, Tcpdump, pero no son tan potentes o carecen de la flexibilidad que nos ofrece el analizador Wireshark.

4.2.3 Anaconda Spyder

La herramienta Anaconda Spyder [Spyder] es uno de los entornos más potentes que hay en la actualidad con soporte para el lenguaje Python y con un entorno de desarrollo interactivo. Debido a su gran cantidad de funciones avanzadas, complementos y bibliotecas hace que esta herramienta sea la más adecuada para la realización del proyecto.

Una de las funciones que más se ha utilizado ha sido la función de depuración. Gracias a esta función se ha podido saber en todo momento si los datos que se han ido obteniendo eran los que realmente esperábamos, y no esperar a que se terminara de ejecutar todo el programa realizado para poder saberlo.

Las bibliotecas que más importancia tienen en el desarrollo de este proyecto son TensorFlow y Keras, por eso, se comentarán el punto siguiente, aunque también se han utilizado otras bibliotecas como NumPy (arreglos multidimensionales eficientes en el cálculo numérico, operaciones con matrices, etc.) y matplotlib (visualización de figuras y datos).

En un principio se iba a utilizar la herramienta MATLAB, ya que era una herramienta que también contaba con librerías de redes neuronales. Se decidió no utilizarla ya que las librerías de Python eran más potentes y variadas.

En la Tabla 4-1 se puede observar una comparación de los puntos más importantes a la hora de elegir Anaconda Spyder frente a MATLAB.

Tabla 4-1 Matlab vs Anaconda Spyder

Universidad	Matlab	Anaconda Spyder (Python)
Utilización durante Grados/Máster	✓	✓
Soporte Sistemas Operativos	✓	✓
Precio	X *Se podría usar licencia UAM	✓ *Licencia gratuita
Librerías dedicadas a las Redes Neuronales	X * Aunque si existen, son más difíciles de manejar	✓
Mejor tratamiento de datos	X	✓ Mas librerías de tratamiento
Últimas actualizaciones Redes neuronales	X *Tardan más en llegar	✓
Decisión final del proyecto	X	✓

4.2.4 Keras y TensorFlow

Keras [KerasB] es una biblioteca diseñada para facilitar el desarrollo de modelos con redes de Aprendizaje Profundo en el menor tiempo posible. Es una biblioteca de alto nivel escrita en lenguaje de programación Python. Keras tiene la capacidad de conectarse y ejecutarse sobre otras librerías como pueden ser TensorFlow, Theano y Microsoft Cognitive Toolkit, siendo para este proyecto ejecutada sobre la primera, TensorFlow. Keras hace más sencillo el desarrollo de los modelos de aprendizaje profundo. Inicialmente se diseñó para la creación en el menor tiempo posible de los modelos redes neuronales profundas, siendo una biblioteca con una gran modularidad y extensibilidad.

TensorFlow [Tensor] es una biblioteca de código abierto y desarrollada por la compañía Google. En un principio fue desarrollada para el uso interno, aunque en noviembre de 2015 decidió publicarlo bajo la licencia de código abierto, por lo que su uso se ha extendido no solamente en productos de la compañía, sino que también para la investigación. El objetivo fundamental es la computación numérica y el aprendizaje automático basado en redes neuronales. Una de sus características más importantes es que TensorFlow puede trabajar en múltiples CPUs y GPUs, lo que hace que los modelos sean más eficientes. Destaca por su gran simplicidad, flexibilidad y la facilidad de uso, contando con actualizaciones

periódicas. TensorFlow entrena y ejecuta redes neuronales profundas para tareas de clasificación, reconocimiento de imágenes, reconocimiento de voz, redes neuronales convolucionales, redes neuronales recurrente entre otras. Está implementado en C++ y Python, siendo la forma más sencilla de utilizarlo mediante una API, Interfaz de programación de aplicaciones, de Python.

4.2.5 Excel

Excel [ExcelM] es una herramienta desarrollada por Microsoft para distintos sistemas operativos. Aunque su principal propósito es la utilización como hoja de cálculo, cuenta con un lenguaje de programación macro llamado Visual Basic.

La gran capacidad y rapidez que nos aporta Excel a la hora de ordenar datos según los parámetros que se establezcan, hace que esta herramienta sea fundamental a lo largo del desarrollo del sistema que se propone de votación.

4.3 Conclusiones

Gracias al conjunto de datos de experimentación y a las herramientas comentadas, se consigue realizar todo el desarrollo *software* del proyecto.

Todas estas herramientas han sido elegidas independientemente y buscando realizar la tarea correspondiente lo más rápida posible y con la máxima sencillez posible. Aunque hay muchas herramientas que podían haber realizado el mismo cometido.

Capítulo 5 Diseño y Desarrollo

5.1 Introducción

En este capítulo se define el diseño que se ha realizado para el sistema. En un primer lugar se explicarán las técnicas utilizadas para el filtrado y adaptación del tráfico de red. Se explicará las diferentes etapas del sistema de red neuronal convolucional utilizado. Además, se comentarán otros métodos de clasificación que se han utilizado.

Por último, se mencionará el algoritmo implementado para mejorar el rendimiento de la clasificación. Siendo la etapa de diseño, la etapa más importante para realizar este proyecto.

Los siguientes apartados se explicarán de orden cronológicos, tal y como se han ido realizando a lo largo del proyecto.

En la Figura 5-1 se puede observar el diagrama de bloques del diseño final del sistema implementado para este proyecto. El primer bloque está en otro color ya que es la única parte del proyecto que no ha sido desarrollada para este proyecto. Todos los demás bloques han sido enteramente desarrollados.

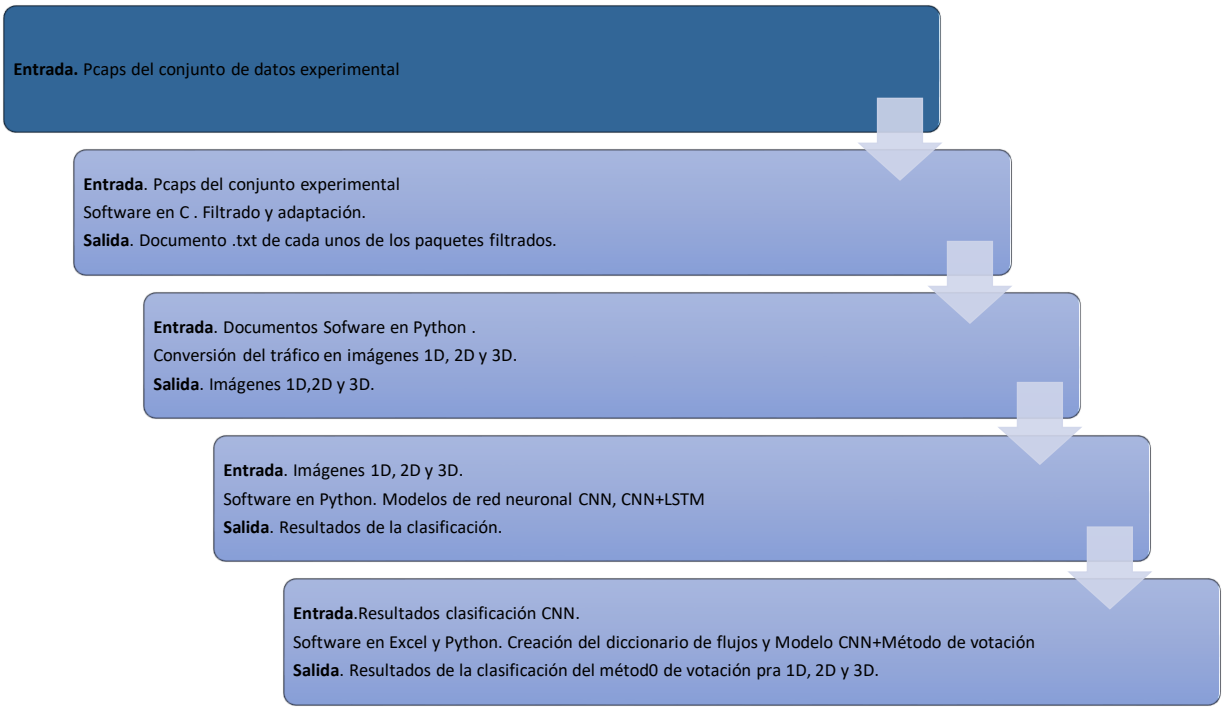


Figura 5-1 Diagrama de bloques entradas/salidas

5.2 Filtrado y adaptación del tráfico

El primer paso que hay que realizar tras obtener el conjunto de datos de experimentación **ISCXVPN2016** es realizar un preprocesado, inspeccionando si el tráfico está realmente filtrado por aplicación, ya que si no lo estuviera no serviría o habría que realizar un nuevo filtrado de estos paquetes. Tras haber realizado la inspección se observa que hay paquetes que no pertenecen a la aplicación, en este momento es cuando se decide filtrar los paquetes que ya se disponen y quedarse con lo que de verdad son de la aplicación, hay que adaptar el tráfico a la aplicación a la que pertenece. Para ello, se toma un paquete que de antemano ya se conozca a la aplicación a la que pertenece y se toman los datos del tipo de tráfico que es (TCP, UDP…). Para cada una de las 42 aplicaciones que se disponen tendremos que conocer el tipo de tráfico que genera. Cuando se tiene claro el tipo de tráfico de cada aplicación, se realiza un nuevo filtrado de cada uno de los archivos pcap del que se dispone. Esto es muy importante, ya que, si no se realizara este filtrado, una vez que se transformara en imagen el paquete y se utilice en la red neuronal, no cumplirá con la estructura visual que las demás imágenes de la aplicación si disponen, por lo que a la hora de clasificar podría ocurrir que o bien se clasificasen mal este tipo de paquetes ya que los clasificaría como la aplicación que realmente es, o haría que los paquetes de otras aplicaciones que se habían filtrado bien se clasificaran de manera incorrecta, lo que provocaría que la tasa de acierto se redujera consideradamente.

Una vez que tenemos todo el tráfico realmente adaptado correctamente, pasaremos a la fase del filtrado específico del tráfico. En esta etapa de filtrado especifico se realiza un nuevo filtrado, pero ya a gusto del propio usuario. Esta etapa podría no realizarse, pero puede tener consecuencias en los resultados obtenidos como ya se verá más adelante.

Para esta parte se realizará el primer desarrollo *software* del TFM. Este desarrollo se ha realizado en el lenguaje de programación C ya que cuenta con varias herramientas que nos permite analizar el tráfico de red. En un primer lugar el usuario deberá introducir el nombre del archivo pcap que se quiere analizar, una vez introducido se mostrará un menú principal en la terminal con diferentes opciones. El objetivo es que el usuario pueda contar con diversas posibilidades y que en un futuro pueda elegir qué tipo de filtrado quiere utilizar dependiendo del tipo proyecto en el que se esté trabajando o de las pruebas que se quieran realizar sin tener que modificar el código constantemente. A su vez, se irá mostrando por pantalla la información más importante y la que más se suele consultar de cada uno de los paquetes del archivo pcap que se había introducido para analizar, a modo analizador de tráfico de red. Alguno de los datos que se muestran son:

- Direcciones MAC [RFC_MAC].
- Direcciones IP [RFC_IP], protocolo de internet.
- Número de Puerto.
- Tipo de protocolo.

- Número de paquete.

Al mostrar estos datos por pantalla, también sirve en el desarrollo para saber si realmente se está realizando de manera correcta el filtrado del paquete, ya que se compararían estos datos con los que proporciona la herramienta Wireshark, lo que es muy útil al principio. A continuación, se comentarán las diferentes opciones que hay dentro del menú citado, habiendo desde las opciones más básicas hasta opciones más complejas:

1. Sin filtrado. Esta es la opción más básica, no se realiza ningún tipo de filtrado, lo único que realizaría seria mostrar información relevante de cada paquete analizado.
2. Filtrado sin cabecera Ethernet. La segunda opción, eliminará de cada uno de los paquetes la cabecera Ethernet.
3. Filtrado por dirección IP. Esta opción se seleccionará si el usuario quiere que solo se adapten el tráfico de una dirección IP concreta, ya sea de origen o de destino.
4. Filtrado por dirección MAC. Opción muy parecida a la anterior, se diferencian en que en este caso el tráfico adaptado será de una dirección MAC concreta, ya sea de origen o de destino.
5. Filtrado por capa de transporte: TCP [RFC_TCP], protocolo de control de transmisión. En esta opción se eliminan todos los paquetes que no pertenezcan a la capa de transporte TCP.
6. Filtrado por capa de transporte: UDP [RFC_UDP], protocolo de datagrama de usuario. De manera análoga, en esta opción se eliminan todos los paquetes que no pertenezcan a la capa de transporte UDP.
7. Filtrado sin cabecera Ethernet ni IP. Opción más conservadora que la siguiente ya que eliminará las cabeceras Ethernet e IP de cada paquete, pero mantendrá la capa de transporte.
8. Filtrado sin cabecera Ethernet, IP ni capa de transporte (TCP/UDP). En la penúltima opción se eliminarán todas las cabeceras Ethernet, IP y la cabecera de la capa de transporte (TCP/UDP)
9. Filtrado por tamaño máximo de paquete. Esta última opción se elegirá si lo que se está buscando no quiere que exceda de un tamaño determinado. Si el tamaño solicita por el usuario es mayor al del paquete que se está examinando, el propio programa rellenará hasta completar el tamaño. Al tratarse de imágenes lo que queremos elaborar, rellenaremos con ceros, que en una imagen se vería representado por el color negro.

Para el proyecto que se ha realizado, la opción de filtrado sin cabecera Ethernet, IP ni capa de transporte (TCP/UDP) es la que se ha escogido. Esto es debido a varios factores, en primer lugar, si nos fijamos en la imagen siguiente que es la cabecera IP, la mayoría de los campos que nos proporciona la cabecera no coinciden entre una conexión a una aplicación y otra

conexión a la misma aplicación. El ejemplo más fácil es la dirección IP origen, si alguien se conecta desde un punto a cualquier aplicación y otra persona se conecta a la aplicación desde otro dispositivo, no van a tener la misma dirección. Es decir, estos campos no nos servirían para saber si dos paquetes de red pertenecen a la misma aplicación. Otro de los factores, es debido a que al realizar las primeras pruebas los resultados obtenidos eran favorables respecto a las otras opciones de filtrado, todas las pruebas y resultados obtenidos se comentaran en detalle en los capítulos posteriores.

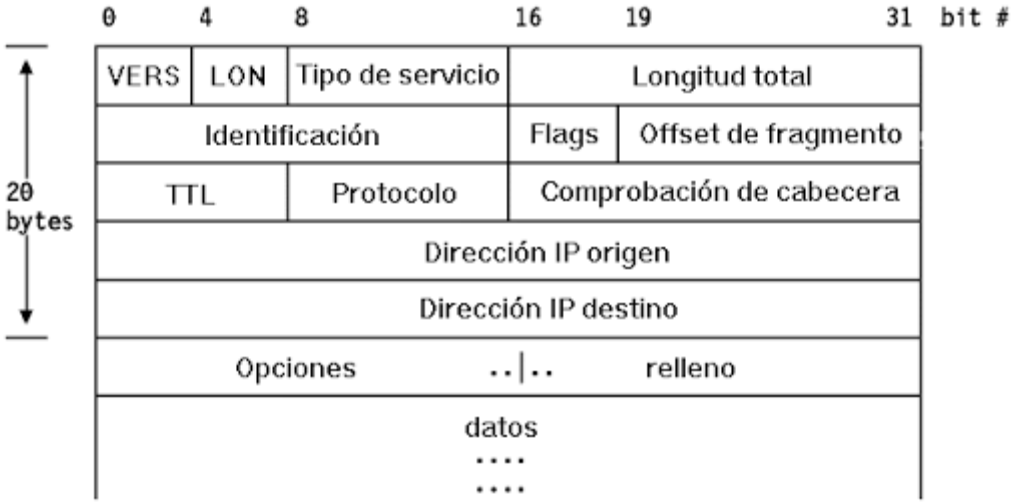


Figura 5-2 Cabecera IP [Formato_ip]

Lo mismo ocurriría con la cabecera TCP y UDP, los campos como puerto de origen o puerto destino no servirían ya que la gran mayoría de las aplicaciones actuales utilizan puertos que están abiertos, esto quiere decir, que una aplicación no siempre tiene porque utilizar el mismo puerto. Otros campos como la longitud, el número de secuencia o suma de control también serán variantes, dependen de muchos otros factores como puede ser la cantidad de datos que se quiere enviar. Todos estos campos se pueden visualizar en la Figura 5-3. Como un paquete no siempre transmite la misma cantidad de datos ni en el mismo orden y son campos que si puede coincidir con otras aplicaciones perfectamente, son campo sin gran relevancia a la hora de crear una estructura.

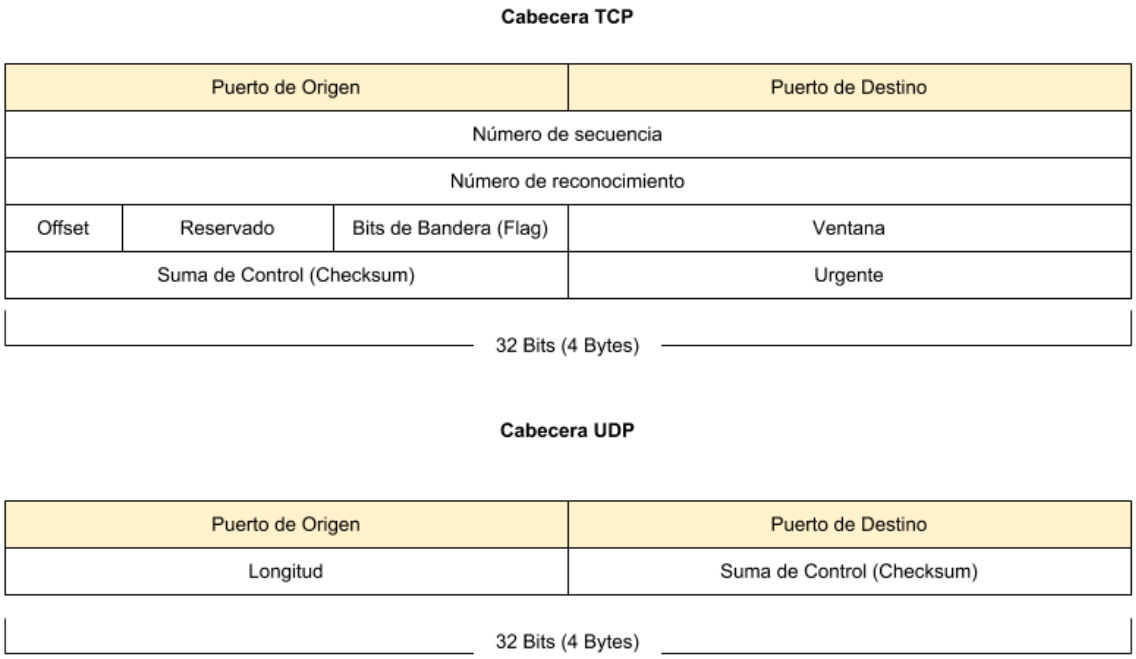


Figura 5-3 Cabecera TCP Y UDP

Con el filtrado lo que se pretende conseguir es eliminar la mayor cantidad de datos que no aporten información realmente útil de la aplicación. Para que a la hora de elaborar las imágenes y utilizarlas con las redes neuronales, estas redes puedan clasificar lo mejor posible.

Una vez que se haya seleccionado la opción deseada, se adaptará todos los paquetes según el tipo de filtrado. A medida que se vayan filtrando los paquetes, paralelamente, se convertirán los datos en binario ya que se necesita para la etapa posterior, transformar los datos en imágenes. Los datos de cada paquete en binario se introducirán en documentos de textos por separado. Cada documento de texto tendrá un nombre especifico, la primera parte corresponderá al nombre de la aplicación a la que pertenece y la segunda parte el número de paquete del archivo del cual se ha extraído (se empezará a nombrar desde el 0).

Por ejemplo:

- Nombre de la aplicación: facebook_audio
- Nombre del archivo: facebook_audio_1a
- Número de paquetes: 7534

El nombre que recibirá el documento de texto del primer paquete del archivo pcap será facebook_audio_1a _00. Como se dispone de 154 archivos pcap y cada uno de ellos contiene miles de paquetes tras ejecutar el programa se habrá creado una carpeta con miles y miles

de archivos de texto, aunque es documentos tendrá un tamaño muy pequeño. En este momento necesitamos que todos documentos estén separados en carpetas diferentes ordenados según la aplicación a la que correspondan. Habría dos opciones, la primera de ella es la que se realizaba al principio cuando se hicieron unas pruebas iniciales, consiste en que manualmente se iba seleccionando los archivos e introduciéndolos en la carpeta de la aplicación. Este método podía servir si había pocos documentos, pero en el momento que ya había una cantidad considerable, se hacía muy lento pudiendo haber hasta errores.

Debido a lo comentado, se decide automatizar este proceso, así en el futuro se ahorraría tanto tiempo como posibles equivocaciones. En esta automatización solo tendría que preocuparse el usuario de crear las carpetas con el nombre de las aplicaciones. Y el programa desarrollado se encargaría de coger el nombre del documento y ver a que carpeta de aplicación pertenece. Tras finalizar el proceso, se habrá conseguido que en cada carpeta estén solo los documentos de textos de la aplicación correspondiente.

Como sabemos el tamaño máximo que van a tener nuestras imágenes, se comentara en la próxima sección, no es necesario que todo el paquete (tras realizar los filtros pertinentes) sea transformado a binario e introducidos en los documentos. Solo se introducirán la cantidad necesaria para crear estas imágenes, así se consigue acelerar el proceso y que los documentos, aunque seguirían siendo de tamaño pequeño, no ocuparan más de la cuenta en el ordenador que se estuviera utilizando.

Cabe destacar que todas estas opciones están enfocadas para conseguir cumplir el objetivo de transformar el tráfico de red, más en concreto, cada paquete de tráfico en imágenes. En ningún momento se ha perseguido realizar un analizador completo de análisis de red, ya que, si se busca eso, hay múltiples opciones de herramientas para utilizar.

5.3 Conversión del tráfico en imágenes

La segunda fase del desarrollo, como se ha ido avanzando en el anterior apartado, es convertir todo el tráfico en imágenes. Para ello se cambiará de lenguaje de programación a Python, en este lenguaje la conversión de datos a imágenes es mucho más sencilla y más rápida.

Este proceso está completamente automatizado, por lo que el usuario no tiene que preocuparse por hacer nada. El *software* desarrollado, convertirá el tráfico en imágenes de una dimensión, dos dimensiones y tres dimensiones (RGB), para poder comparar los resultados obtenidos tras utilizar los modelos de red neuronal desarrollados.

Dependiendo del tamaño de los paquetes, el programa podrá convertir los datos a un tamaño de imagen u otro. Como decisión técnica de desarrollo se ha decidido que todas las imágenes tengan el mismo tamaño dentro del tipo de imagen. En la Tabla 5-1 se observan los diferentes tipos de imágenes:

Tabla 5-1 Tipos de Imágenes

Tipo de Imagen	Nº de bytes	Tamaño Imagen	Tráfico
1D	784	1x784	VPN No VPN
2D	784	28x28	VPN No VPN
3D	729	9x9	VPN No VPN
3D	1331	11x11	VPN No VPN

Como se puede observar en la Tabla 5-1, las imágenes tipo 1D y 2D son las imágenes en escala de grises y tienen el mismo tamaño en bytes. En las imágenes Figura 5-4 y Figura 5-5 se puede visualizar una representación de este tipo de imágenes.

Figura 5-4 Representación Imagen 1D

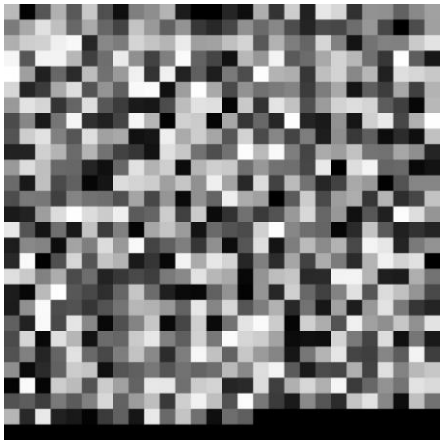


Figura 5-5 Representación Imagen 2D

Ambas imágenes son representaciones de las imágenes obtenidas, se ha tenido que realizar un reescalado para que se puedan apreciar en este documento. Si se pusiese el tamaño original no se conseguiría apreciar como son las imágenes.

En la Tabla 5-1 se indica que las imágenes en 3D tienen dos representaciones, la primera con un tamaño de 9x9 pixeles que sería el primer tamaño posible por debajo de los 784 bytes y

la segunda representación, tiene un tamaño de 11x11 pixeles, con 1331 bytes, aunque se podría haber elegido otro tamaño cualquiera mientras fuera superior. Se decide realizar dos representaciones ya que las imágenes 3D como son imágenes RGB, es decir, con los colores rojo, verde y negro, necesitan más datos para poder representarlas y se quiere comparar los resultados con diferentes cantidades de datos para ver como varia.

A continuación, en la Figura 5-6 se puede ver la representación genérica de una imagen en 3D. Como ocurre con las imágenes 1D y 2D, se hace un reescalado para que se pueda apreciar algo. Por eso, tampoco se ponen las dos representaciones, 9x9 y 11x11, ya que al haberse aumentado manualmente el tamaño no habría diferencias visuales.

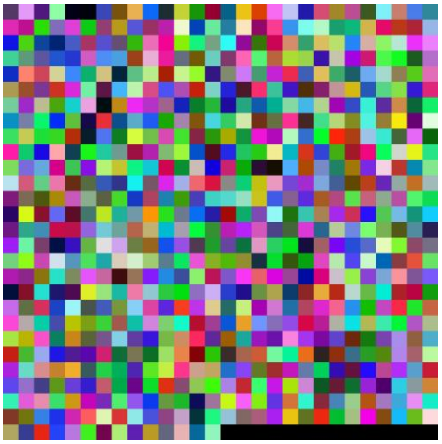


Figura 5-6 Representación Imagen 3D

Como se puede ver en la Figura 5-5 y la Figura 5-6 la última parte de la figura, la que se encuentra más abajo del todo, está completamente negra. Esto se debe, como ya se explicó con anterioridad, a que el tamaño del paquete no es lo suficientemente grande para el tamaño de imagen que se ha selecciona por eso se rellena hasta el final con el color negro. No todas las imágenes tendrán esta peculiaridad.

Aunque el tamaño en pixeles puede parecer pequeño, es suficiente para que la red neuronal pueda distinguirlas.

5.4 Clasificación de imágenes

En esta sección se comentarán las principales características que comparten los tres métodos que se has propuesto en el proyecto. El desarrollo de la clasificación de imágenes se realizará en el lenguaje de programación Python y se utilizarán tanto redes neuronales convolucionales como redes neuronales recurrentes. Para cada uno de los modelos creados, se realizarán las pruebas tanto para imágenes de 1D, 2D y 3D. Como no tienen las mismas longitudes, se tendrá especial cuidado con las funciones utilizadas debido a que no todas soportan las mismas dimensiones de entrada.

En el estado del arte se ha podido comprobar que las redes neurona dan muy bueno resultado a la hora de clasificar imágenes, así que se intentará crear un buen modelo para que clasifique bien las imágenes creadas que contienen el tráfico de red.

A la hora de clasificación de las imágenes necesitamos que el número de imágenes por clase esté balanceado, no serviría de nada que en una categoría hubiera 10000 imágenes y en otra solamente 50. En este caso la probabilidad de que clasificara bien la primera categoría de imágenes sería superior porque al separar por conjunto de entrenamiento, validación y test, se entrenaría mucho más esa categoría, la red neuronal obtendría más patrones y característica, por lo que sería más normal que se consiguieran mejores resultados. Tampoco es necesario que se tengan las mismas imágenes exactamente, con que sea una cantidad parecida, valdría.

Para realizar el balanceado, antes de introducir las imágenes por la red neuronal, se seleccionan aleatoriamente un número determinado de imágenes que se ha establecido. Lo primero que se realizaría es ir categoría por categoría viendo si superan el umbral de imágenes, en el caso de este proyecto se ha establecido en un mínimo de 3500 imágenes y un máximo de 5000 imágenes. Todas las categorías que se comprobaran que no tenían esta cantidad de imágenes se eliminaría, ya que se ha considerado que dejaría de estar balanceado el número de imágenes. Con el resto que cumpliera el umbral, se seleccionaría, aleatoriamente las imágenes. Gracias a este proceso conseguiríamos que todas las categorías estuvieran balanceadas respecto al umbral que se ha establecido de imágenes.

Sabiendo que hay un total de 42 categorías y una media de 4250 imágenes por categoría, la cantidad de imágenes que va a tener que procesar el sistema de clasificación es un total de 178500 imágenes, que es una cantidad bastante importante.

Las 42 categorías que se han clasificado en este proyecto son:

1. facebook_audio
2. facebook_chat
3. facebook_video
4. ftps_down
5. ftps_up
6. hangouts_audio
7. hangouts_chat
8. hangouts_video
9. scp
10. scpDown
11. scpUp
12. sftp
13. sftpDown
14. sftpUp
15. skype_audio
16. skype_chat
17. skype_file
18. skype_video
19. torFacebook
20. torGoogle_t
21. torTwitter
22. torVimeo
23. torYoutube
24. vimeo
25. voipbuster
26. vpn_email
27. vpn_facebook_audio
28. vpn_facebook_chat
29. vpn_ftps
30. vpn_hangouts_audio
31. vpn_hangouts_chat
32. vpn_icq_chat
33. vpn_sftp
34. vpn_skype_audio
35. vpn_skype_chat
36. vpn_skype_files
37. vpn_spotify
38. vpn_vimeo
39. vpn_voipbuster
40. vpn_youtube
41. youtube
42. youtubeHTML

El siguiente paso es dividir las imágenes en los subconjuntos deseados. Para ello se dividirá el conjunto de datos en un 80% para entrenamiento y un 20% para test. A su vez, el conjunto

de entrenamiento lo subdividiremos en 80% para el set de entrenamiento real y 20 % para validación. Por lo que quedaría 64% para entrenamiento, 20% test y 16% validación como muestra la Figura 5-7.

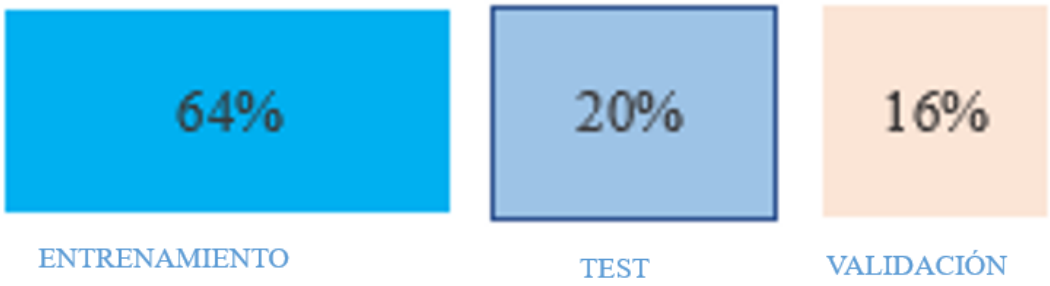


Figura 5-7 Distribución seleccionada

Una vez que tenemos los tres subconjuntos, es hora de pasarlo por los modelos creados. El número de épocas es el número de veces que el conjunto de datos de entrenamiento circula por el modelo creado. Cuantas más épocas se ejecuten, más mejorará el modelo, hasta cierto punto. Después de ese punto, el modelo dejará de mejorar durante cada época. Para los modelos desarrollados, se fijará el número de épocas en 6. En los siguientes apartados, se explicarán con detalle las características de los tres modelos.

Una vez que se hayan pasado los datos de entrenamiento por el modelo, habrá que compilar el modelo. Para la compilación del modelo es necesario definir tres parámetros: optimizador, pérdida y métrica.

El optimizador controla la tasa de aprendizaje. Se usará el llamado *'adam'* como nuestro optimizador con un ritmo de aprendizaje de $1e{-}3$. *'Adam'* es generalmente un buen optimizador ya que solventa el problema con la fijación de la ratio de aprendizaje. El optimizador 'adam' ajusta el ritmo de aprendizaje a lo largo del entrenamiento, es decir, se basa en una ratio de aprendizaje adaptativo. Si los parámetros están muy dispersos la ratio de aprendizaje aumentará.

Se utilizará 'categorical_crossentropy' para la función de pérdidas. Esta función de pérdidas se utiliza para la categorización de una sola etiqueta. Esto es cuando sólo se aplica una categoría para cada dato. En el caso de este proyecto, una imagen no puede pertenecer a dos aplicaciones a la vez.

Como métrica se utilizará 'Accuracy', exactitud en castellano. Este tipo de métrica se utiliza para la clasificación de modelos, gracias a este parámetro se conseguirá la relación entre el número de predicciones correctas y el número total de muestras de entrada. La métrica solo funciona bien cuando las clases están balanceadas. Si no estuviera balanceado, la fórmula de la Ecuación 5-1 Accuracy no serviría. Por ejemplo, consideremos que hay un 96% de muestras de clase A y un 4% de muestras de clase B en el conjunto de entrenamiento. Entonces el modelo puede obtener fácilmente una precisión de entrenamiento del 98% simplemente prediciendo cada muestra de entrenamiento perteneciente a la clase A.

$$Accuracy = \frac{NumeroCorrectoPredicciones}{NumeroTotalPredicciones}$$

Ecuación 5-1 Accuracy

Como el conjunto de datos de este proyecto se ha balanceado, es una buena métrica para utilizar.

Otra métrica utilizada es la matriz de confusión ya que se quiere conocer el rendimiento por categoría. La matriz de confusión, como su nombre indica, proporciona una matriz como salida y describe el rendimiento completo del modelo. La Ecuación 5-2 Accuracy Matriz de confusión utilizada ya no depende de si las categorías están o no balanceadas.

$$Accuracy = \frac{VerdaderosPositivos + FalsosNegativos}{NumeroTotalEjemplos}$$

Ecuación 5-2 Accuracy Matriz de confusión

Hay 4 términos importantes:

- Verdaderos positivos: Los casos en los que se predicen como acierto y el resultado real también lo era
- Verdaderos Negativos: Los casos en los que se predicen como acierto y el resultado real era falso.
- Falsos positivos: Los casos en los que se predicen como acierto y el resultado real era negativo.
- Falsos negativos: Los casos en los que se predicen como negativo y el resultado real era positivos.

Por último, se realizará el informe de clasificación. Este informe muestra una representación de las principales métricas de clasificación por clase, dando una intuición más profunda del comportamiento del clasificador sobre la exactitud global.

Las métricas de este informe se definen en términos de verdaderos y falsos positivos, y verdaderos y falsos negativos. Usando esta terminología, las métricas se definen de la siguiente manera:

- *Precision*. La precisión es la capacidad de un clasificador de no etiquetar casos positivos que en realidad son negativos. Para cada clase se define como la relación entre los verdaderos positivos y la suma de los verdaderos y falsos positivos. A continuación, se puede ver la formula.

$$Precision = \frac{TP}{TP + FP}$$

Ecuación 5-3 Precision

- *Recall* (Exhaustividad). Es la capacidad de un clasificador para encontrar todos los casos positivos. Para cada clase se define como la relación entre los verdaderos positivos y la suma de los verdaderos positivos y los falsos negativos. A continuación, se puede ver la formula.

$$Recall = \frac{TP}{TP + FN}$$

Ecuación 5-4 Recall

- *F1-score*. La puntuación de la F1 es una media ponderada de precisión y exhaustividad tal que la mejor puntuación es 1 y la peor es 0. En general, las puntuaciones de la F1 son inferiores a las medidas de precisión, ya que incorporan la precisión y la exhaustividad en sus cálculos. A continuación, se puede ver la formula.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Ecuación 5-5 F1

- *Support*. Es el número de casos reales de la clase en el conjunto de datos especificado.

Con todas estas medidas se pretende obtener el rendimiento real de los modelos creados, os resultados serán expuestos en el Capítulo 6 Pruebas y resultados. Una vez terminadas de explicar todas las características que comparten los tres modelos, se pasará a explicar la estructura de los modelos por separado.

5.4.1 Redes neuronales convolucionales

El primer método propuesto y en el que se basarán los demás métodos es el sistema de red neuronal convolucional, CNN. Como el modelo creado se utiliza para imágenes en 1D, 2D y 3D, habrá diferencias en las entradas, pero la estructura es la misma.

El modelo de red neuronal por el cual atraviesan el flujo de datos es el siguiente:

1. Entrada. La imagen del conjunto de datos de entrenamiento es introducida, una vez que ha sido procesada.
2. Convolución. Por cada canal de la imagen de entrada se tiene 32 filtros. Se empleará para regular, una etapa de *batch normalization*. La salida de la primera capa convolucional son 32 imágenes a escala de grises.

3. A la salida de la capa se aplica una etapa *max pooling* de 2x2, es utilizado para reducir dimensiones de la salida de la red, en la siguiente etapa se aumenta la profundidad de la capa convolucional, consiguiendo así características más complejas.
4. Convolución. En esta capa se utilizan 64 filtros. La salida por lo tanto son 64 imágenes a escala de gris. También se aplica una etapa de *batch normalization* como pasaba en la primera convolución.
5. A la salida de la capa se le aplica un *max pooling* de 2x2
6. Convolución. En la tercera y última capa de convolución se utilizan también 64 filtros. La salida nuevamente serán 64 imágenes a escala de gris. Se aplica una etapa de *batch normalization*.
7. A la salida de la capa se le aplica un *max pooling* de 2x2.
8. Tras realizar la capa anterior, se le realiza una etapa flatten para conseguir una sola dimensión.
9. Función de activación Softmax.
10. Etapa *Dense* con 128 filtros.
11. Por último, otra etapa *Dense* con el tamaño del número de clases (42) filtros.

La función de activación que se ha utilizado por el modelo en las capas ha sido: *ReLU*. Con esta capa se consigue introducir la no linealidad de los datos, ya que los datos del mundo real no son lineales.

Para este aprendizaje se ha utilizado un mecanismo por lotes (*batch*), de tamaño 64 con los que se consigue modificar los pesos.

A lo largo del desarrollo del modelo se probaron diferentes estructuras, con más y menos capas de convolución, número de filtros, diferentes funciones de activación. Pero la estructura final es la anteriormente mostrada. Otro modelo que se utilizó y tuvo también muy buenos resultados es igual que el modelo descrito, pero quitando los pasos 6 y 7, es decir, la última capa de convolución.

5.4.2 Redes neuronales convolucionales y recurrentes

La estructura de este modelo consiste en intercalar las capas del modelo anterior con capas de una red neuronal recurrente, más en concreto la red llamada LSTM.

La arquitectura de red de este modelo CNN + LSTM:

1. Entrada. La imagen del conjunto de datos de entrenamiento es introducida, una vez que ha sido procesada.
2. Convolución. Por cada canal de nuestra imagen de entrada se tiene 32 filtros. Se empleará para la regularización, una etapa de *batch normalization*. La salida de la primera capa convolucional son 32 imágenes a escala de grises.
3. A la salida de la capa se aplica una etapa *max pooling* de 2x2 es utilizado para reducir dimensiones de la salida de la red, en la siguiente etapa se aumenta la profundidad de la capa convolucional, consiguiendo así características más complejas.

4. Capa LSTM (50). El LSTM transforma la secuencia de vectores en un único vector de tamaño 50, que contiene información sobre toda la secuencia. En el LSTM, el modelo aprende qué información almacenar en la memoria a largo plazo y de qué deshacerse.
5. Convolución. En esta capa se utilizan 64 filtros. Por lo que la salida son 64 imágenes a escala de gris. También se aplica una etapa de *batch normalization* como pasaba en la primera convolución.
6. A la salida de la capa se le aplica un *max pooling* de 2x2
7. Capa LSTM (25). Transformación de la secuencia de vectores en un único vector de tamaño 25.
8. Convolución. En la tercera y última capa de convolución se utilizan también 64 filtros. La salida nuevamente serán 64 imágenes a escala de gris. Se aplica una etapa de *batch normalization*
9. A la salida de la capa se le aplica un *max pooling* de 2x2.
10. Capa LSTM (25). Transformación de la secuencia de vectores en un único vector de tamaño 25.
11. Tras realizar la capa anterior, se le realiza una etapa *flatten* para conseguir una sola dimensión.
12. Función de activación *Softmax*.
13. Etapa *Dense* con 128 filtros.
14. Por último, otra etapa *Dense* con el tamaño del número de clases (42) filtros.

En este modelo a cada salida de una capa convolucional se aplica la red LSTM, se intenta que la red obténgala información temporal de las imágenes. Se quiere que con el modelo LSTM se superen a los otros modelos, ya que las imágenes tienen componentes temporales y este tipo de red aprende de las dependencias a largo plazo. La capacidad del LSTM para olvidar, recordar y actualizar la información lo lleva un paso adelante de los demás modelos.

Para este método no se ha utilizado un aprendizaje por lotes ya que si lo usáramos se tomarían las imágenes aleatoriamente y se perdería la ventaja de utilizar las redes neuronales recurrentes. La función de activación, como ya pasaba en el anterior modelo, que se ha utilizado ha sido: *ReLU.*

5.4.3 Redes neuronales convolucionales y método de votación propuesto

En este modelo se utiliza la red neuronal convolucional explicada en el apartado 5.4.1 Redes neuronales convolucionales por lo que no se volverá a explicar.

Lo primeo que hay que realizar es un diccionario en un documento de texto. Este diccionario consiste en agrupar las imágenes por flujos. Para ello se ha desarrollado un *software* en lenguaje C, que introduce en el documento por línea tanto la quíntupla de datos del paquete como el nombre de la imagen. Cuando se tiene estos datos, mediante una macro de Excel creada se ordena por flujos. Un flujo es un conjunto de paquetes de información que comparten quíntupla y se transmiten de forma unidireccional. Como los datos viajan a través de flujos, todos los paquetes de un flujo pertenecerán a la misma aplicación.

Se llama quíntupla al conjunto de datos de la cabecera de un paquete que se compone por:

- Dirección IP de origen
- Dirección IP de destino
- Puerto TCP/UDP de origen
- Puerto TCP/UDP de destino
- Protocolo de la capa de aplicación

En la Figura 5-8 se puede ver recuadrados los campos que pertenecen a la quíntupla de un paquete, en este caso la cabecera es TCP, aunque ocurriría lo mismo con UDP.

Figura 5-8 Quintupla [Quin_es]

Ahora que tenemos el diccionario de flujos creados y la salida de la red neuronal, se pueden ver qué imágenes de las que se ha clasificado lo han hecho correctamente o erróneamente.

El método de votación que se ha propuesto y desarrollado se basa en tomar una imagen de las que se ha clasificado y buscarla en el diccionario de flujos. Cuando se haya encontrado, se buscará cada imagen del flujo para ver ene que categoría a clasificado. Es aquí donde entra la votación, se ha decido que solo las primeras 20 imágenes del flujo serán las que importen en la votación. De estas 20 imágenes, cada una tendrá una ponderación diferente, de la primera a la décima el voto contará doble y de la undécima a la vigésima el voto contará normal. Una vez que se hayan votado las imágenes, la categoría que tenga más votos será la ganadora y se cambiará la categoría de la imagen que se había buscado en un principio. Este proceso se realizaría mediante bucle para poder realizar la misma operación con todas las demás imágenes que habían sido clasificadas. Este proceso requiere de bastante tiempo ya que tiene que comprobar muchas imágenes.

Se ha tomado una serie de decisiones al realizar este método de votación. La primera, es la ya comentada de la ponderación de las imágenes. Si solo hubiera una imagen el flujo, la imagen se volvería clasificar como lo había clasificado el modelo de red neuronal. Otro suceso que pudiera ocurrir es que la votación saliera empate, si esto ocurre y alguna de las categorías que están implicadas en el empate es la categoría inicial de la CNN, se clasificará con esa categoría. Si ninguna fuera la categoría, se elegiría una al azar.

Gracias a las votaciones, se consigue obtener la moda del flujo. La idea de solo tener en cuenta las primeros 20 imágenes viene dada del trabajo de la Universidad de Valladolid [Martin17], explicado en el Estado del Arte. Aunque ellos decían que cualquier número superior a 5-15 se obtenían resultados parecido, se ha decidido dejar un margen hasta 20. También de este estudio ha venido la idea de que los primeros paquetes tengan una votación superior. Todas estas ideas se comprobaron durante la validación del trabajo, para ver si eran realmente válidas.

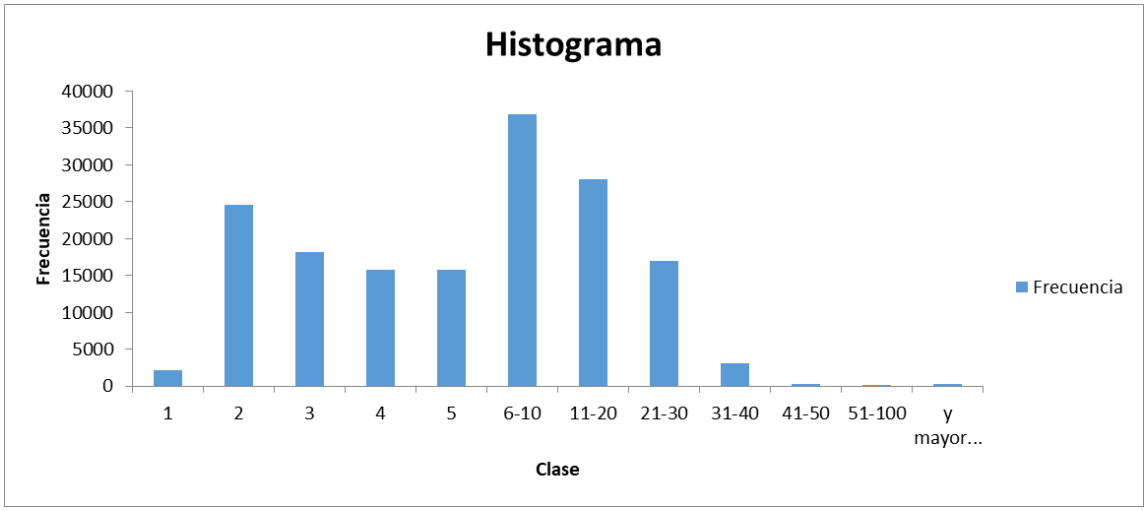


Figura 5-9 Histograma Flujos

En la Figura 5-9 se observa un histograma de los flujos de paquetes que se han tratado. El eje X representa el número de paquetes y el eje Y la frecuencia, cantidad que se repite. Como ya se había comentado con anterioridad, los datos se transmiten mediante flujos y en el caso de este proyecto la mayoría de los flujos tiene una cantidad de entre 6 y 10 paquetes. Los tamaños grandes de flujos se pueden deber a las aplicaciones de video, ya que en una misma sesión se transmite muchos paquetes con el contenido del video para poder así visualizarlo.

5.5 Conclusiones

Toda la etapa de diseño y desarrollo se ha tenido que hacer conjuntamente con la de Pruebas y resultados para que se pudiera ver los resultados y ver si había que modificar el diseño de los modelos creados.

Durante el proyecto se han realizado múltiples desarrollos que se han tenido que ir modificando porque no se obtenían o bien los resultados deseados o se quería ver si otros modelos mejoraban los resultados ya obtenidos.

Esta etapa es fundamental en el proyecto ya que sin un buen diseño y desarrollo no se podría llegar a ninguna solución, por lo que es la tapa donde más tiempo se ha invertido. Los modelos mostrados en esta sección has sido con los que mejores resultados se ha clasificado.

Tanto en la fase de filtrado y adaptación del tráfico como en la de conversión del tráfico en imágenes el usuario puede elegir diferentes opciones para trabajar o se utilizarán los valores

por defecto. Esta funcionalidad puede servir para que en un futuro se hagan pruebas con otro tamaño de imágenes, otros filtros de imágenes…

Capítulo 6 Pruebas y resultados

6.1 Introducción

En este capítulo se mostrarán las distintas pruebas que se han realizado para el Proyecto de Fin de Máster, también se describirán los resultados más relevantes obtenidos. Se han realizado múltiples pruebas con el propósito de ver el correcto funcionamiento de todos los desarrollos que se han realizado.

En este capítulo se podrá ver por fin si las redes neuronales convolucionales sirven realmente para clasificar el tráfico que circula por las redes.

6.2 Descripción de las pruebas realizadas y resultados intermedios

En este apartado se explicarán todas las pruebas realizadas durante el desarrollo del proyecto. Se irán comentando en el mismo orden cronológico al que se fueron desarrollado para así poder seguir la línea temporal. También se comentarán los resultados de las pruebas que no fueron definitivas en los modelos creados.

En primer lugar, se realizaron diversas pruebas para ver si el *software* implementado para el filtrado del tráfico de red funcionaba correctamente. Para ello se utiliza los datos obtenidos por pantalla y se compara con los obtenidos en la herramienta Wireshark. Con esta comprobación se consigue que en todo momento se sepa que los paquetes analizados se les ha realizado los filtros seleccionados.

En el proceso de conversión del tráfico de red en imágenes se han realizado también una serie de pruebas. Las primeras han sido para ver si el tamaño correspondía con lo que se había predefinido inicialmente, posteriormente se comprobaba que las imágenes se habían creado con los colores correspondientes (1D y 2D en escala de grises y 3D en RGB). Cuando ya se había comprado estos detalles, se intentó ver si las imágenes de una misma categoría tenían alguna estructura que pudiera reconocer una persona, esta comprobación no tuvo buenos resultados ya que no se encontraron estructuras que se pudieran diferenciar.

Aunque la clasificación de imágenes 1D da buenos resultados, se realiza también la clasificación para imágenes 2D y 3D ya que se quiere aprovechar la computación paralela que se puede conseguir con estas imágenes En un caso real de utilización de estos sistemas el número de bits por segundo que se pueden explotar paralelamente, para imágenes 2D y 3D, es mayor, por lo que se conseguiría clasificar más rápido.

Hasta ahora todas las pruebas habían sido relativamente rápidas de comprobar, el resto de las pruebas sí que llevarían más tiempo para poder comprobarlas. Una vez balanceadas todas las clases que se quieren clasificar, se tiene que comprobar categoría a categoría que tienen un número de imágenes parecido, ya que es necesario para poder utilizar las métricas diseñadas para evaluar los modelos de red neuronal. Lo mismo habría que hacer con el conjunto de entrenamiento, validación y test, se comprueba que siguen las dimensiones que se establecieron. A la hora de desarrollar el modelo se realizan incontables pruebas, las primeras del todo era ver si las dimensiones de los datos en la entrada y salida eran los esperados. Posteriormente se empezó a construir la red neuronal, las primeras versiones eran solo con una capa convolucional y sin ningún tipo de función de activación, como era de esperar como no se tenía muchos conocimientos a la hora de desarrollar modelos no se obtuvieron buenos resultados, tal es así que con este modelo tan precario solo se obtuvo un 0,4% de acierto, es decir no clasificaba. Fue en este momento en el que tuvo que revisar todo el desarrollo anterior, se comprueba que las categorías no estaban bien balanceadas. Ya con categorías bien balanceadas, el resultado mejora llegando al 4.5% de acierto en las imágenes 1D y valores muy similares con las otras imágenes. Era un gran avance ya que ya empezaba a clasificar. Corregido también algunos errores de dimensiones, se obtienen en 1D un 10%, en 2D un 8% y por último en 3D un 6.5% de acierto.

En los siguientes pasos se fueron añadiendo capas y la función de activación, poco a poco se fueron aumentado los porcentajes de acierto consiguiendo un 38.8% con 42 categorías y si se bajaba el número de categorías a 12 se mejoraba la clasificación al 69% para imágenes en 3D. También se elevó el conjunto de datos que se podían evaluar, hasta el momento solo se había estado probando con un conjunto pequeño para que no fuera un proceso bastante largo sin saber si realmente se iba a conseguir clasificar este tipo de imágenes. Para imágenes 2D con 42 categorías clasificaba bien el 26%, en cambio con una red menos compleja, las imágenes 1D, con 32 categorías se llegaba a un acierto del 76%. Se llega a un punto que reduciendo las capas de convolución, pero normalizando los datos en cada capa realizada, se llegan a unos resultados en el que las imágenes 1D se consigue una tasa de acierto del 93.2%, las imágenes 2D un 93% y las imágenes 3D un 92.5%. En este punto que se tienen ya unos valores de acierto superior al 90% es la hora de ver si con otros filtros o tamaños de las imágenes se obtenían mejores resultados. Se empieza a probar con imágenes más grandes ya que se pensaba que al tener más datos podría clasificar mejor, pero esto no ocurría. Esto es debido a que no todos los paquetes del tráfico que circula contienen cantidades de datos muy grandes por lo que se tenía que rellenar muchas de las imágenes con el color negro. Al tener muchas imágenes la parte del final todo en negro, la red neuronal lo empezaba a confundir como una característica propia de una sola categoría y clasificaba muchas de las veces esa categoría. Esto se pudo comprobar gracias a la matriz de confusión. Como no se obtuvieron los esperados, se decide hacer lo contrario, utilizar imágenes más pequeñas. Con imágenes más pequeñas tampoco se consiguen mejorar resultados, esto se puede deber a que la red neuronal no conseguía obtener patrones para clasificar las imágenes.

Se prueba a ir añadiendo cabeceras de trasporte y red, pero ni mejora ni empeora mucho la clasificación. A medida que se iban añadiendo cabeceras los resultados empeoraban muy ligeramente. Esto se debe a que se introducirían datos que no aportan mucha información de una aplicación, ya que serían diferentes para cada imagen de una misma aplicación. No empeora mucho la clasificación ya que si comparas la cantidad de bytes que aportan estas cabeceras con el resto de dato, es una suma muchas veces casi insignificante.

Como no se consiguieron mejorar los resultados con ninguna de estas pruebas, se decidió empezar con las pruebas de la red neuronal recurrente, en el caso de este proyecto la red CNN + LSTM. Para este modelo como se incorporaba todo el sistema LSTM después de cada capa de convolución, las dimensiones no cuadraban al principio. Una vez solucionado este problema, se consigue llegar más rápido a resultados relativamente buenos. Con el modelo explicado en los capítulos anteriores, se obtenían resultados muy dispares. Se realizan también las mismas pruebas que para la red CNN, pero no se consigue estabilizar los resultados.

Teniendo en cuenta los resultados obtenidos, para el método de votación solo se realizarían pruebas con la red CNN. Como el método de votación tiene que utilizar el diccionario de flujos, lo primero que se tuvo que comprobar mediante un pequeño conjunto de paquetes donde había varios flujos era ver si los flujos estaban separados correctamente y no se había colado ningún paquete en un flujo que no fuera el suyo. El método de votación utiliza pesos dependiendo de la posición en el flujo, al principio del todo se había diseñado para que la primera imagen del flujo con 3 votos, las imágenes de la 2 a las 10 contaran 2 votos y el resto contara únicamente un voto. Se realizaron pruebas donde todas las imágenes contaban el mismo número de votos, pero se obtenían resultados un poco inferiores. Hasta ese momento la votación se realizaba sobre todas las imágenes del flujo por lo que era un proceso bastante lento, ya que si había muchas imágenes en un flujo tenían que votar todas y las ponderaciones del principio no afectaban en la mayoría de los casos. Se decide bajar el número máximo que se tendría en cuenta en la votación a 10 imágenes y se realiza varias pruebas con diferentes ponderaciones. se llegó al sistema de votación con el que se adquiere un 95.5% de acierto en el mejor de los casos. Ajustando el número de imágenes para clasificar e igualando la ponderación a dos votos hasta la décima imagen se consigue la máxima optimización del sistema de votación. Reduciendo el número de imágenes a 20 se consigue a su vez que el sistema sea mucho más rápido si se compara al primer prototipo que utilizaba todas las imágenes de los flujos.

6.3 Descripción de los resultados obtenidos

Una vez establecidos los modelos de las redes neuronales que se van a utilizar para clasificar las diferentes aplicaciones del tráfico que circula por internet, y la métrica empleada para esto, en esta sección se mostrarán los resultados obtenidos. Se ha dividido esta sección dependiendo el modelo que se haya utilizado mostrando los resultandos finales

de 1D, 2D y 3D. En el Apéndice A. Matrices de confusión pueden ver las matrices de confusión de los modelos CNN y CNN+ Método de votación para los tres tipos de imágenes.

6.3.1 Resultados obtenidos CNN

En este primer subapartado se verán los resultados del modelo CNN utilizado.

Tabla 6-1 1D CNN Resultados

1D-CNN	precision	recall	F1-score	support
exactitud	0.94	0.94	0.94	33800
macro avg	0.93	0.92	0.92	33800
weighted avg	0.94	0.94	0.94	33800

Tabla 6-2 2D CNN Resultados

2D-CNN	precision	recall	F1-score	support
exactitud	0.93	0.94	0.94	33800
macro avg	0.91	0.92	0.91	33800
weighted avg	0.95	0.94	0.94	33800

Tabla 6-3 3D CNN Resultados

3D-CNN	precision	recall	F1-score	support
exactitud	0.94	0.94	0.95	33800
macro avg	0.93	0.91	0.91	33800
weighted avg	0.94	0.94	0.94	33800

6.3.2 Resultados obtenidos CNN + LSTM

Con este modelo se han obtenido los peores resultados, ya que no eran estables. Si se ejecutaba una vez podía dar resultados mejores que el modelo que solo utiliza CNN, pero en la siguiente ejecución la precisión podía bajar bastante. Esto ocurrirá para los tres tipos de imágenes.

Tabla 6-4 1D CNN + LSTM Resultados(1ªEjecución)

1D- CNN + LSTM	precision	recall	F1-score	support
exactitud	0.92	0.92	0.92	33800
macro avg	0.92	0.93	0.92	33800
weighted avg	0.94	0.92	0.93	33800

Tabla 6-5 1D CNN + LSTM Resultados (2ªEjecución)

1D- CNN + LSTM	precision	recall	F1-score	support
exactitud	0.80	0.80	0.80	33800
macro avg	0.80	0.81	0.80	33800
weighted avg	0.82	0.81	0.81	33800

Tabla 6-6 2D CNN + LSTM Resultados(1ªEjecución)

2D-CNN + LSTM	precision	recall	F1-score	support
exactitud	0.91	0.91	0.91	33800
macro avg	0.91	0.92	0.91	33800
weighted avg	0.92	0.91	0.91	33800

Tabla 6-7 2D CNN + LSTM Resultados(2ªEjecución)

2D-CNN + LSTM	precision	recall	F1-score	support
exactitud	0.65	0.66	0.65	33800
macro avg	0.66	0.66	0.66	33800
weighted avg	0.65	0.66	0.65	33800

Tabla 6-8 3D CNN + LSTM Resultados(1ªEjecución)

3D- CNN + LSTM	precision	recall	F1-score	support
exactitud	0.91	0.90	0.90	33800
macro avg	0.90	0.90	0.90	33800
weighted avg	0.91	0.91	0.91	33800

Tabla 6-9 3D CNN + LSTM Resultados(2ªEjecución)

3D- CNN + LSTM	precision	recall	F1-score	support
exactitud	0.80	0.80	0.80	33800
macro avg	0.80	0.81	0.80	33800
weighted avg	0.82	0.80	0.80	33800

6.3.3 Resultados obtenidos CNN + Método de votación propuesto

Por último, se verán los resultados obtenidos del modelo CNN + Método de votación propuesto. Este es con el que mejor resultados se obtienen para todo tipo de imágenes.

Tabla 6-10 1D CNN + Método de votación Resultados

1D- CNN + Método de votación	precision	recall	F1-score	support
exactitud	0.95	0.96	0.94	33800
macro avg	0.94	0.93	0.92	33800
weighted avg	0.94	0.94	0.94	33800

Tabla 6-11 2D CNN + Método de votación Resultados

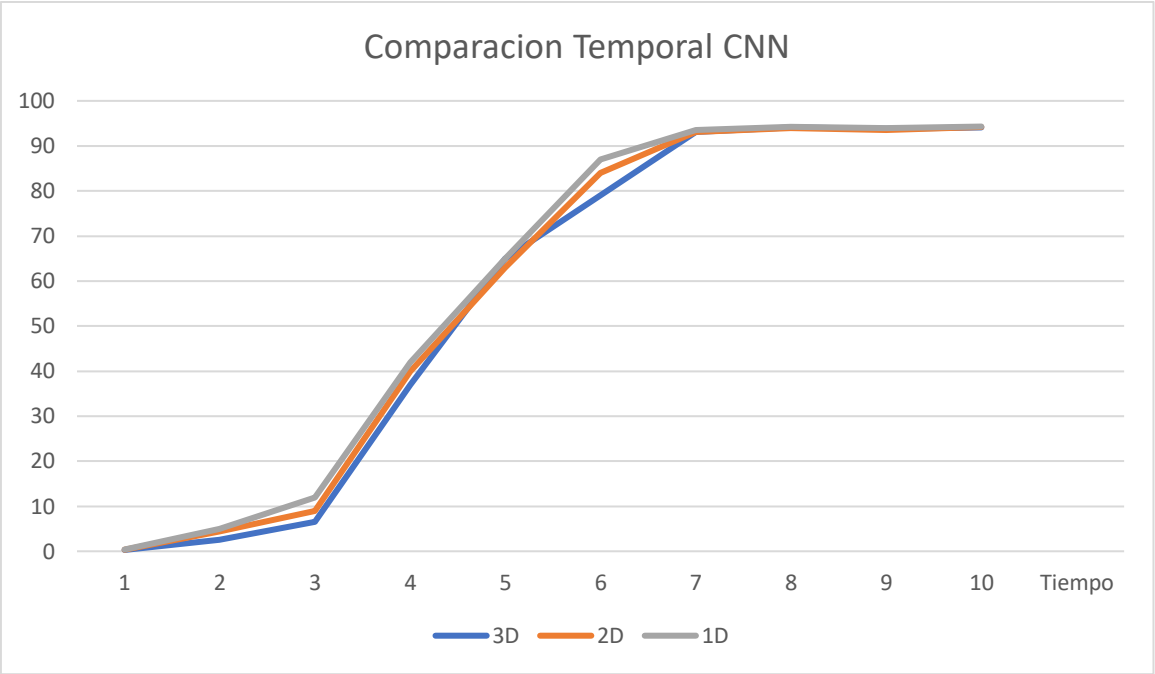
2D- CNN + Método de votación	precision	recall	F1-score	support
exactitud	0.94	0.94	0.94	33800
macro avg	0.91	0.91	0.91	33800
weighted avg	0.94	0.94	0.94	33800

Tabla 6-12 3D CNN + Método de votación Resultados

3D- CNN + Método de votación	precision	recall	F1-score	support
exactitud	0.94	0.94	0.95	33800
macro avg	0.91	0.91	0.92	33800
weighted avg	0.95	0.95	0.95	33800

6.4 Comparación de resultados

A continuación, en las siguientes tres gráficas se verá cómo ha ido evolucionando el porcentaje de acierto de cada uno de los modelos. El eje y representa el porcentaje de Exactitud y el eje x las progresiones temporales. Como están balanceadas las categorías teniendo un número de imágenes muy parecido, los resultados de exactitud ya sea de la Fórmula de Exactitud o la Fórmula de Exactitud de la Matriz de Confusión dan resultados muy parecidos, se diferencian por decimales.

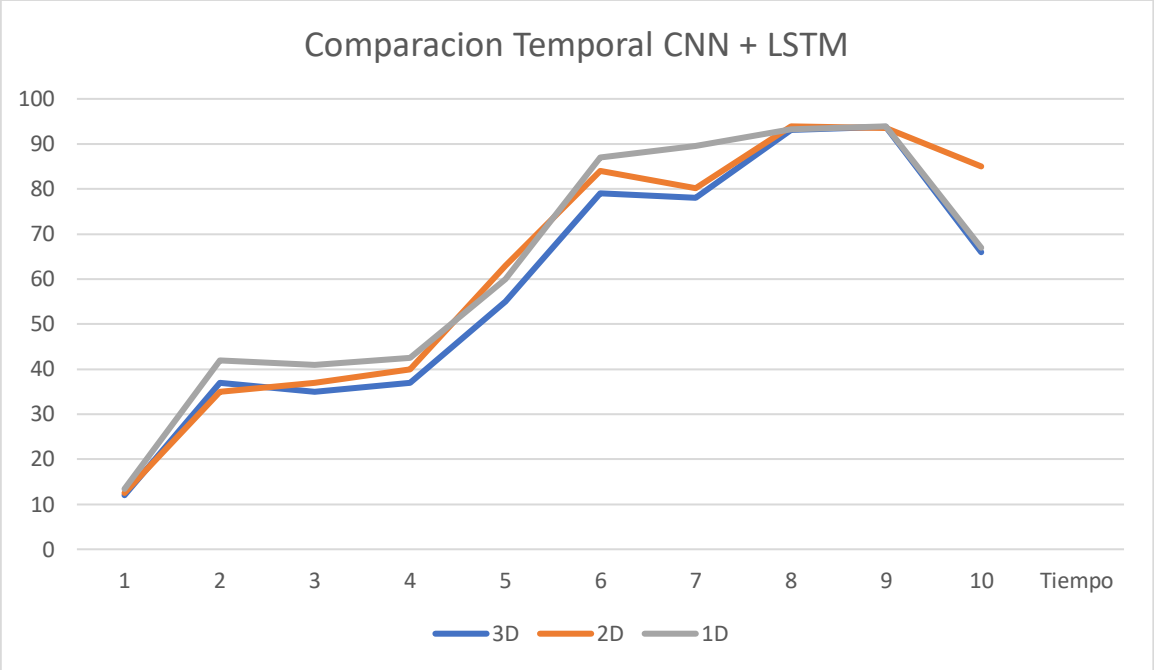


Gráfica 6-1 Comparación Temporal CNN

La primera gráfica que se comentará es la Gráfica 6-1Se puede observar que hay tres etapas muy diferenciadas, la primera de ellas es la inicial, en esta etapa el crecimiento es lento ya que no se manejaba muy bien los conocimientos para crear un buen modelo de red neuronal y también se tenía muchos fallos. La segunda etapa, es la intermedia, esta etapa destaca por el crecimiento tan fuerte que tiene. Esto se debe a que se empezaba a tener más conocimientos, se elaboró un modelo más profundo con capas de activación, normalización a las salidas de las capas. Y en último lugar, se encuentra la etapa en la que estando ya en porcentajes altos (por encima del 90% de acierto) se produce un estancamiento. En este estancamiento no se consigue mejorar el modelo CNN.

También se puede ver que las imágenes de 1D siempre se han obtenido mejores tasas de acierto, mientras que las imágenes de 2D y 3D se iban intercalando el segundo y tercer puesto dependiendo de las capas que tuviera la red neuronal o las funciones utilizadas. En algunos momentos el crecimiento era más exponencial y en otros casos más lineal.

La diferencia que hay entre una ejecución y otra, sin realizar ningún cambio, es totalmente normal ya que los porcentajes de acierto no cambian mucho. Ocurre para los tres tipos de imágenes.

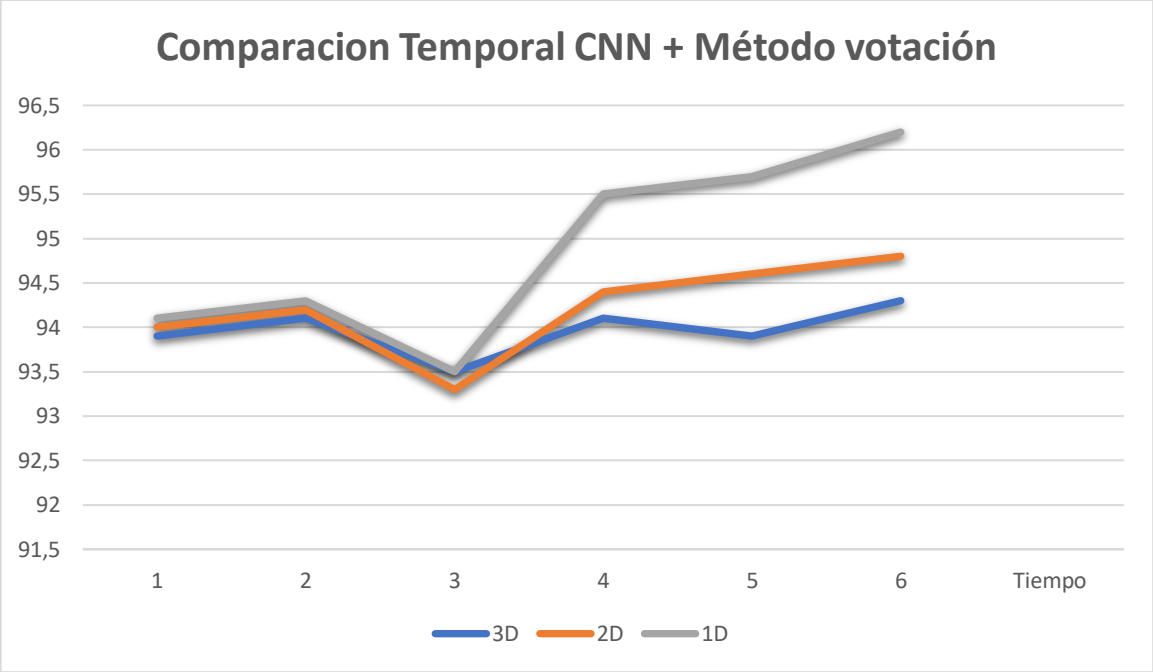


Gráfica 6-2 Comparación Temporal CNN + LSTM

En la Gráfica 6-2 se comentará el modelo CNN + LSTM. Si comparamos el crecimiento inicial que tiene esta red, es muy superior al del modelo CNN. Tiene una explicación muy sencilla, como este modelo se crea a partir del modelo CNN y no desde 0, ya se había conseguido eliminar algunos errores y se empezaba a elaborar un modelo más complejo, por eso el crecimiento es mucho más rápido.

Una vez terminado ese crecimiento inicial, el modelo va creciendo muy paralelamente al anterior modelo, habiendo etapas que supera al modelo CNN gracias a la componente temporal que es analizado por la red LSTM. Hay un momento, cuando se llegan a los porcentajes del 80 %, que este modelo deja de estar estable. En una iteración puede conseguir un valor de acierto muy alto, y en la siguiente sin tocar nada descender un 20%. A medida que se modifica la red y se conseguían resultados más altos en una iteración, más era el porcentaje que podía varias en siguientes iteraciones. Por ejemplo, para las imágenes de 1D, en una iteración se conseguía un 93,7% en la siguiente un 93,3% y en una siguiente iteración un 66%. Si la diferencia fuera como en las dos primeras iteraciones, sería algo normal ya que depende de las imágenes que se clasificaran se obtendrían mejores o peores resultados, pero como de una diferencia de un 0,6% se podía pasar a una diferencia del 27% y no se consigue estabilizar estas diferencias, se decide dejar de desarrollar este modelo para centrarse más en otros donde la diferencia sí que sea estable.

En este modelo no se comparará por el tipo de imágenes porque todo depende de la ecuación y al no ser estable, en algún momento puede tener mejor resultados las imágenes 3D en otros las 1D. Si solo se tuviera en cuenta la primera parte, cuando el modelo era estable, pero se obtenían unos resultamos muy por debajo, las imágenes 2D y las imágenes 1D eran las que mejor clasificaban, siendo as imágenes 2D un poquito superior.



Gráfica 6-3 Comparación Temporal CNN + Método de votación

En la Gráfica 6-3 se explicará cómo ha sido la evolución del modelo CNN + Método de votación. Este modelo empieza con los resultados obtenidos de la mejor configuración CNN, una vez que se utiliza el método de clasificación con todas las imágenes del flujo, se mejoran los resultados obtenido un 0,3% de media. Como el proceso requería demasiado tiempo, se decide bajar el número de imágenes a 10, pero los resultados bajan un 0.5%. Se eleva el número de imágenes que se utilizaría para evaluar a 20 y ajustando los pesos de las imágenes se consigue un aumento en los porcentajes de acierto. Por último, tras ajustar el peso de las votaciones una vez más, se consigue elevar y conseguir el porcentaje máximo de acierto aun 96,2% en imágenes 1D, 94,75% en las imágenes 2D y 94.4% las imágenes 3D. Las imágenes en 1D sufrieron un cambio significante desde el inicio hasta concretar el modelo definitivo de votación. En cambio, las imágenes 2D y 3D, aunque si que mejoran los resultados, no lo realizan de una manera tan grande.

Con este método la dispersión que hay entre una ejecución y otra, sin realizar ningún cambio, es normal, los porcentajes de acierto no difieren mucho. En el caso de las imágenes 1D, en el mejor de los casos se consigue 96,2%, en la siguiente ejecución un 95,8% y en otra ejecución un 96,1% de acierto, lo que son valores muy cercanos, no como pasaba con el método que utiliza LSTM.

Una vez que ya se ha visto la evolución temporal y se han comparado los modelos dependiendo del tipo de imagen que se utilizara, se va a proceder a comparar entre los métodos, no se tendrá en cuenta el segundo modelo CNN + LSTM.

Como no sería justo compararlos en base a la evolución temporal debido a que el último método parte del modelo CNN, solo se comprarán los resultados respecto al nivel de acierto.

Se ha podido ver que las imágenes en 1D son las que mejor clasifica en los modelos descritos. En el modelo CNN la diferencia de utilizar un tipo de imagen u otra solo se diferencia en décimas, pero en cambio en el método de votación, si se utiliza las imágenes 1D hay una diferencia de un 1.5% respecto a utilizar las otras imágenes.

Si ahora se compara los resultados del método CNN con el método CNN + Método de votación, sin duda alguna el método de votación obtiene mejores resultados, mejorando casi un 2%en el caso de las imágenes 1D. En el caso de las imágenes 2D y 3D no mejoran tanto un porcentaje tan grande los resultados, pero aun así mejora un 0.7% y 0,6% de media respectivamente.

6.5 Conclusiones

En este capítulo se ha podido ver la gran cantidad de pruebas que se han realizado para obtener los mejores resultados posibles. El realizar pruebas también es un gran mecanismo para solucionar posibles errores que se pueden haber cometido durante el desarrollo. En cuanto a los resultados conseguidos, el método CNN + Método de votación propuesto es el que mejores resultados ha logrado, consiguiendo hasta un 96.2% de acierto en la clasificación de 42 aplicaciones. El método de votación propuesto ha sido un éxito ya que ha conseguido mejorar los resultados de la CNN un 2%, porcentaje bastante alto teniendo en cuenta que ya se habían alcanzado muy buenos resultados.

En contra posición se encuentra el método CNN + LSTM, un modelo donde se había depositado grandes esperanzas debido a su componente temporal. Como no se consiguió estabilizar los resultados se tuvo que abandonar esta idea.

Por concluir, aunque estos resultados obtenidos son realmente bastante buenos comparándose con otros proyectos, se debería intentar conseguir un porcentaje aún mayor para así ser una herramienta fiable a la hora de clasificar tráfico en redes.

Capítulo 7 Conclusiones y trabajos futuros

7.1 Conclusiones

En este Trabajo de Fin de Máster, se ha seguido un proceso de análisis, desarrollo y optimización de un modelo de red neuronal convolucional, con el objetivo de clasificar el tráfico que circula por las redes, utilizando para ello su carga útil, que puede encontrarse cifrada. Gracias a una serie de tareas se conseguido cumplir con los objetivos mercados desde un inicio.

Tras realizar un estudio del estado del arte, se ha conseguido entender muchos de los conceptos que se han utilizado en este proyecto. También la gran necesidad de crear un nuevo mecanismo, diferente a los métodos tradicionales de clasificación de tráfico de red, debido a que gran cantidad de las aplicaciones que utilizan internet a día de hoy son capaces de cifrar los datos o utilizar otros métodos para no ser reconocidos, por lo que no se pueden utilizar los métodos tradicionales para clasificar el tráfico.

El estado de arte ha sido de gran ayuda para aportar ideas a la fase de diseño y desarrollo. Se llega a la conclusión que la utilización de redes neuronales convolucionales puede ser un gran medio para conseguir un buen método para clasificar el tráfico de red gracias a la gran capacidad de clasificar imágenes. La única incógnita que quedaba era ver si sería capaz de clasificar imágenes que no representan nada para el ojo humano. Con una buena configuración de la red neuronal y múltiples pruebas de estructura, la duda se despeja. Se ha obtenido resultados con un acierto del 94% utilizando solo la red neuronal. Aunque otros estudios con el mismo conjunto de datos de experimentación han obtenido mejores resultados, no clasifican la misma cantidad de aplicaciones, así que no se puede hacer una comparación real.

En este punto, una vez que se logra muy buenos resultados de clasificación con el modelo de red neuronal convolucional desarrollado, es el momento en el que se decide intentar mejorar un poco más los resultados. Para ello se desarrolla un sistema de votación propio que utiliza los resultados de la red neuronal convolucional. Se consigue mejorar los resultados de clasificación un 2%, llegando a un acierto del 96% que es un porcentaje bastante alto si se tiene en cuenta el gran número de aplicaciones que se clasifica.

Aunque estos resultados logrados son verdaderamente altos, los estudios sobre técnicas de mejora no se deberían finalizar ya que una buena clasificación del tráfico de red puede ayudar a la hora de gestionar las redes o posibles ataques.

En conclusión, este trabajo ha fomentado el uso de conocimientos adquiridos durante el máster en diferentes asignaturas, en particular la utilización de distintos elementos para

trabajar con redes neuronales y aspectos del tráfico en redes, a través de un ejemplo práctico real. Este proyecto multidisciplinario agrupa distintas áreas de la Ingeniería de Telecomunicación, telemática y sonido e imagen, y ha sido de gran utilidad las asignaturas cursadas durante el máster para el desarrollo de este TFM, sino se hubieran cursado estas asignaturas hubiera sido de una complejidad mayor la realización.

7.2 Trabajo futuro

A continuación, tras ver los resultados que se han obtenido en este proyecto se propondrá posibles mejoras para así conseguir mejores resultados:

- Automatizar todo el proceso. Aunque actualmente todo el proyecto está semiautomatizado, hay partes en las que el usuario tiene que cambiar de plataforma para poder continuar. Lo que se podría realizar para mejorar este proyecto, es que desde la entrada de los pcaps para filtrarlos hasta la salida después de hacer la clasificación estuviera automatizado, realizándose en un solo lenguaje de programación o creando un script que una vez finalizado un proceso ejecute directamente el siguiente.

- Acelerar el proceso. El proceso de clasificación es relativamente lento, aunque depende mucho de la potencia del ordenador que se utilice, aun así, se podría buscar técnicas que mejoren estos tiempos para que las técnicas sean realmente útiles y se puedan utilizar como una herramienta real.

- Aumentar las aplicaciones. Aunque en el conjunto de datos de experimentación que se ha utilizado está compuesto por 42 aplicaciones, muchas de ellas no son utilizadas hoy en día por un gran número de personas. Si se aumentaran el número de aplicaciones con otros conjuntos de datos de experimentación, se podría ver qué resultado se obtienen tras un entrenamiento previo.

- Obtener mejores resultados. Si bien se han obtenido grandes resultados con el modelo de redes neuronales y el método de votación propuesto, se podrían ver otro tipo de redes que utilizándolas pudieran mejorar los resultados obtenidos, bien conjuntamente con el modelo ya creado o independientemente.

Apéndice A. Matrices de confusión

En este apéndice se mostrarán las matrices de confusiones correspondientes al modelo CNN y al modelo CNN + Método de Votación para los tres tipos de imágenes, 1D, 2D y 3D. Las matrices de confusión son:

- **CNN 1D**

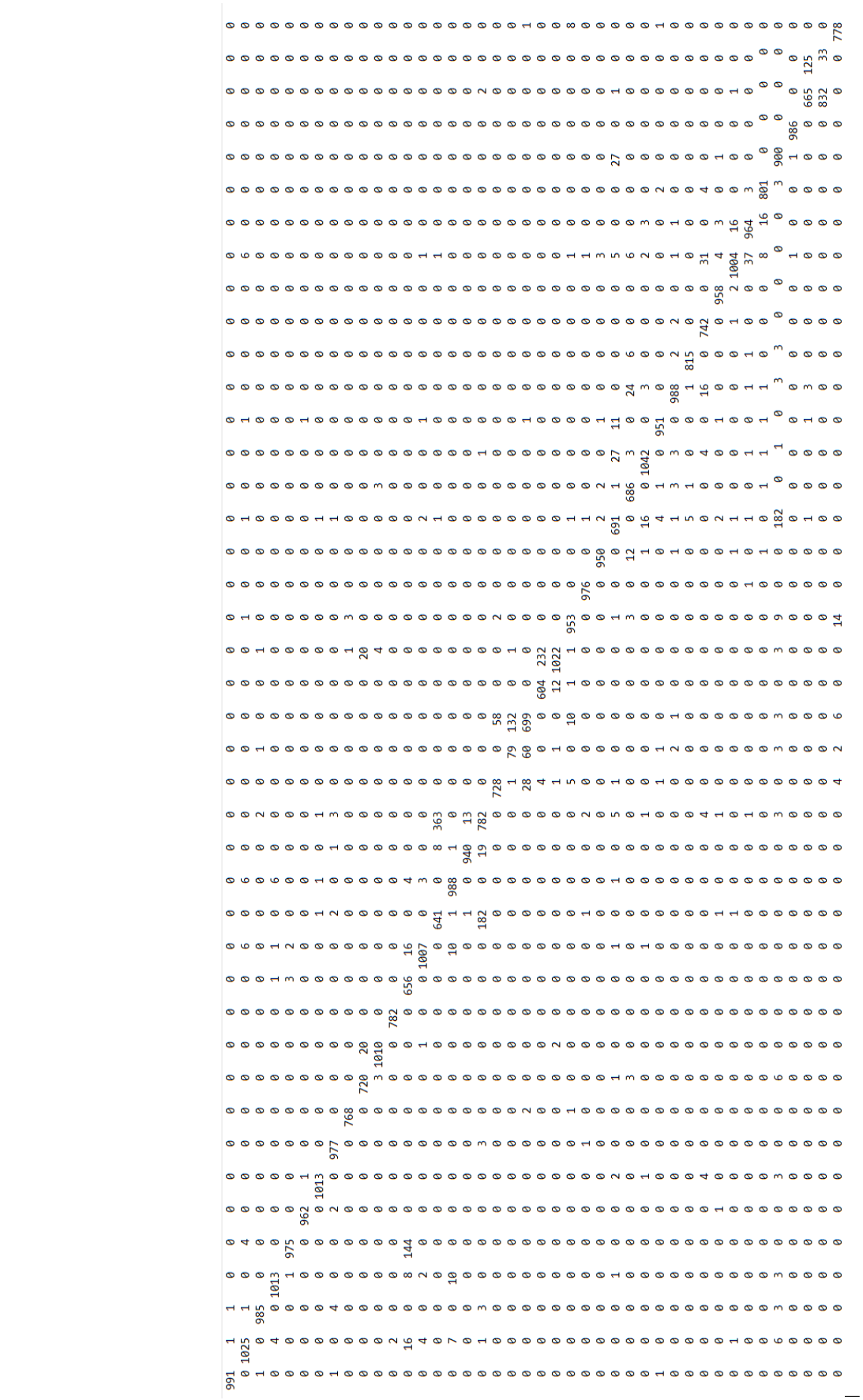


Figura 0-1 Matriz Confusión CNN 1D

- **CNN 2D**

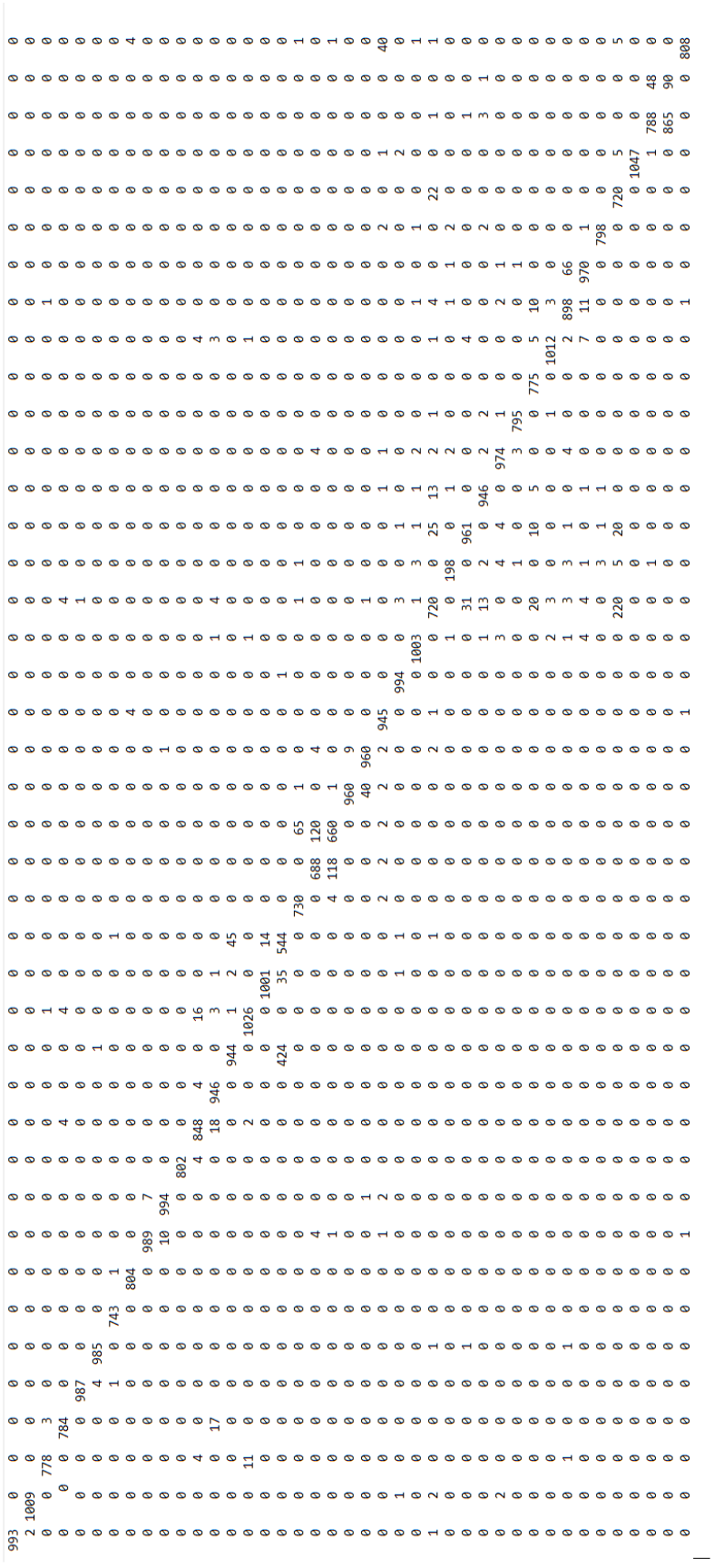


Figura 0-2 Matriz Confusión CNN 2D

- **CNN 3D**

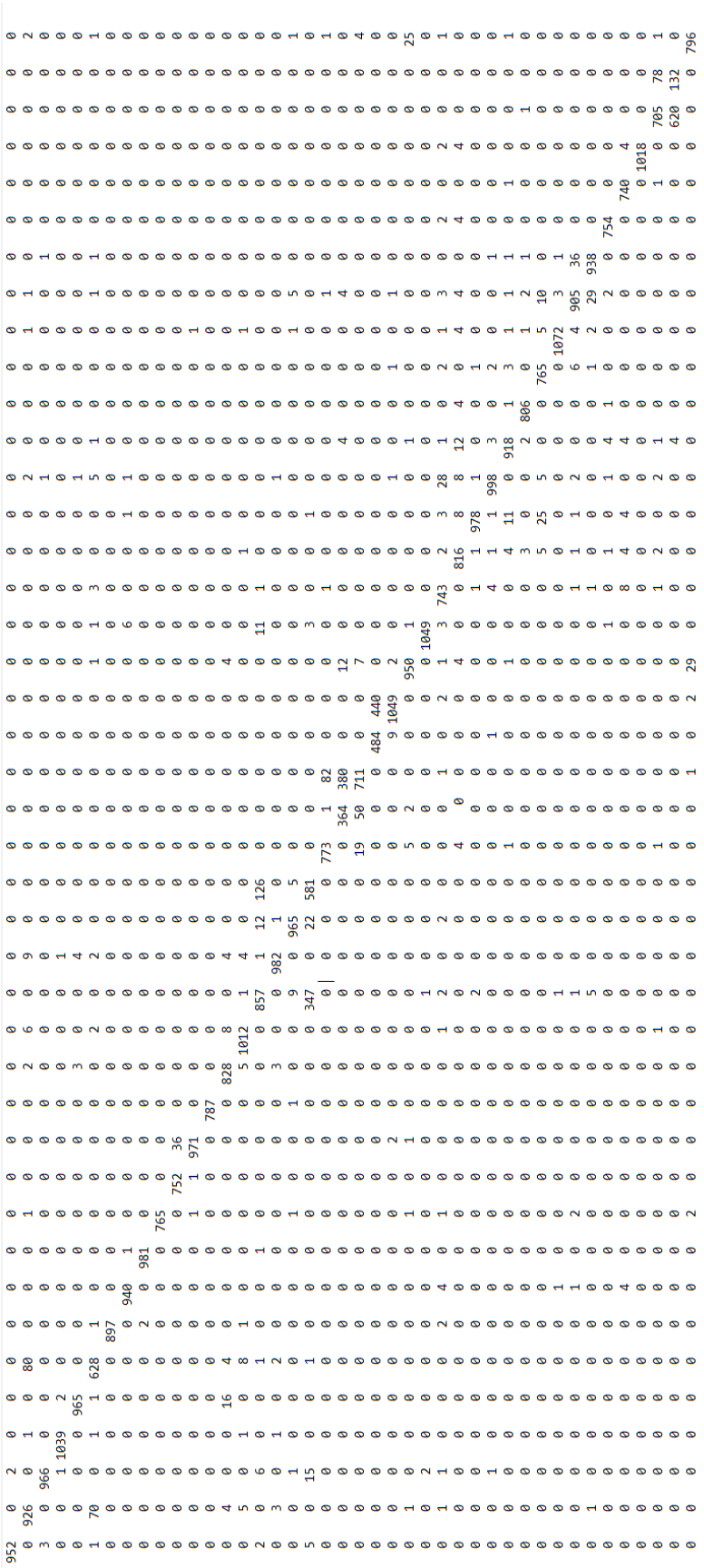


Figura 0-3 Matriz Confusión CNN 1D

- **CNN 1D + Método de Votación**

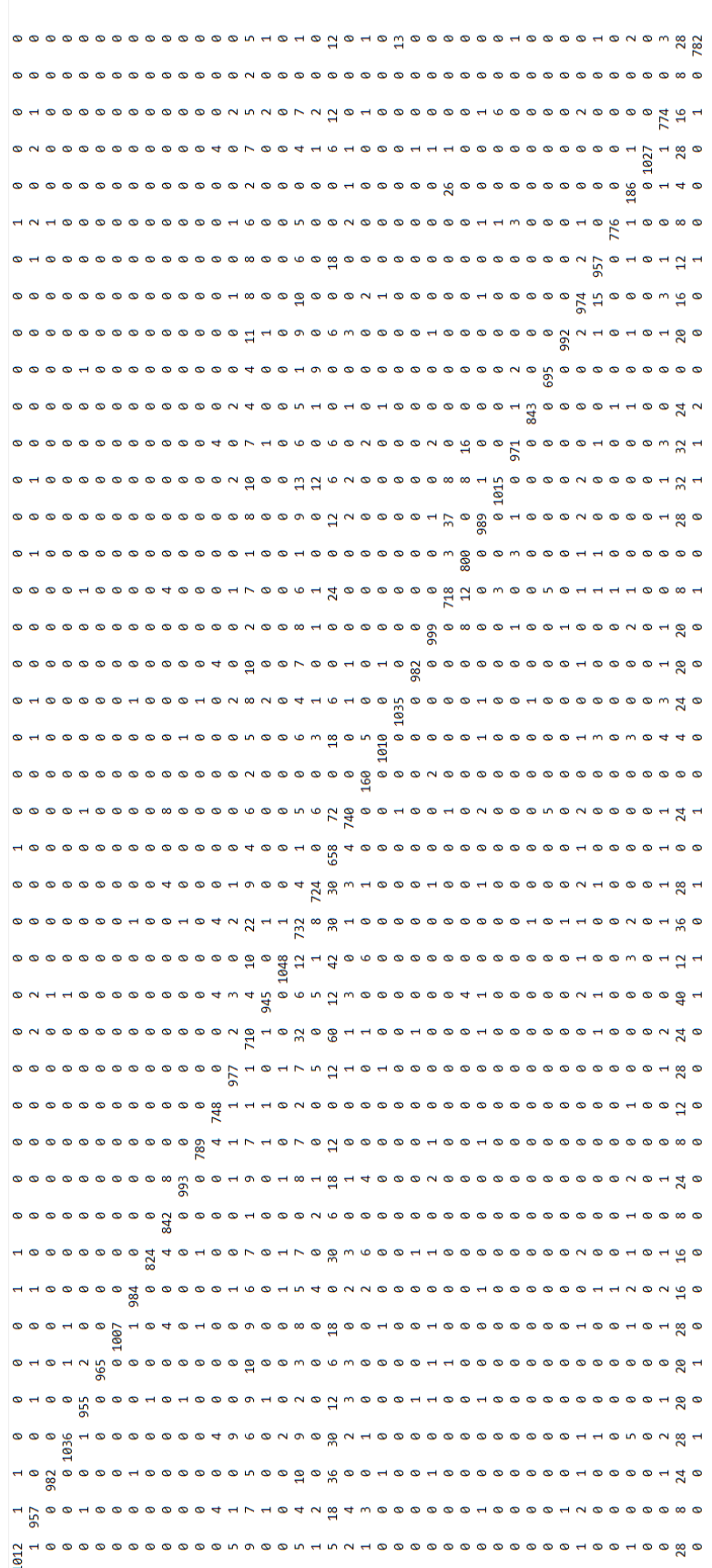


Figura 0-4 Matriz Confusión CNN 1D + Método de votación

- **CNN 2D + Método de Votación**

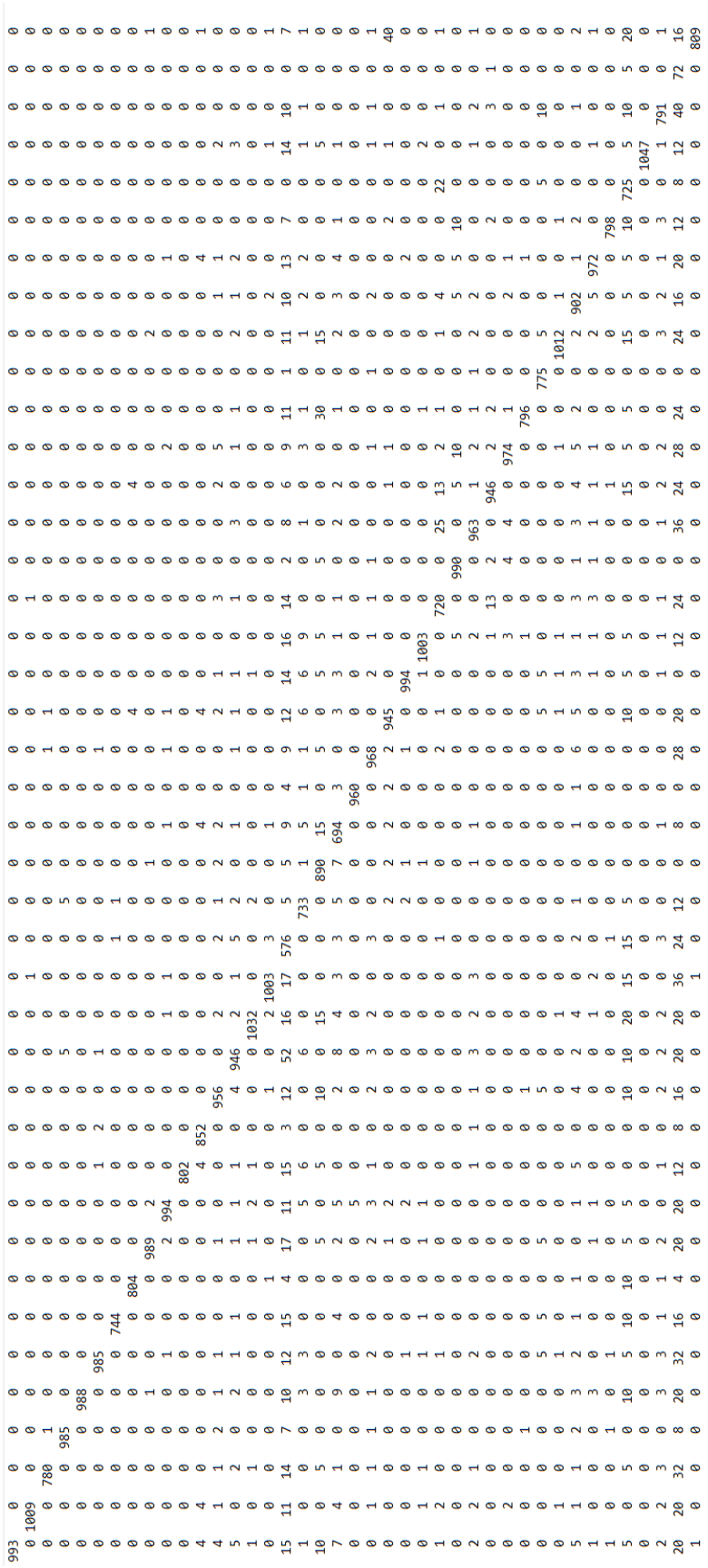


Figura 0-5 Matriz Confusión CNN 2D + Método de votación

- **CNN 3D + Método de Votación**

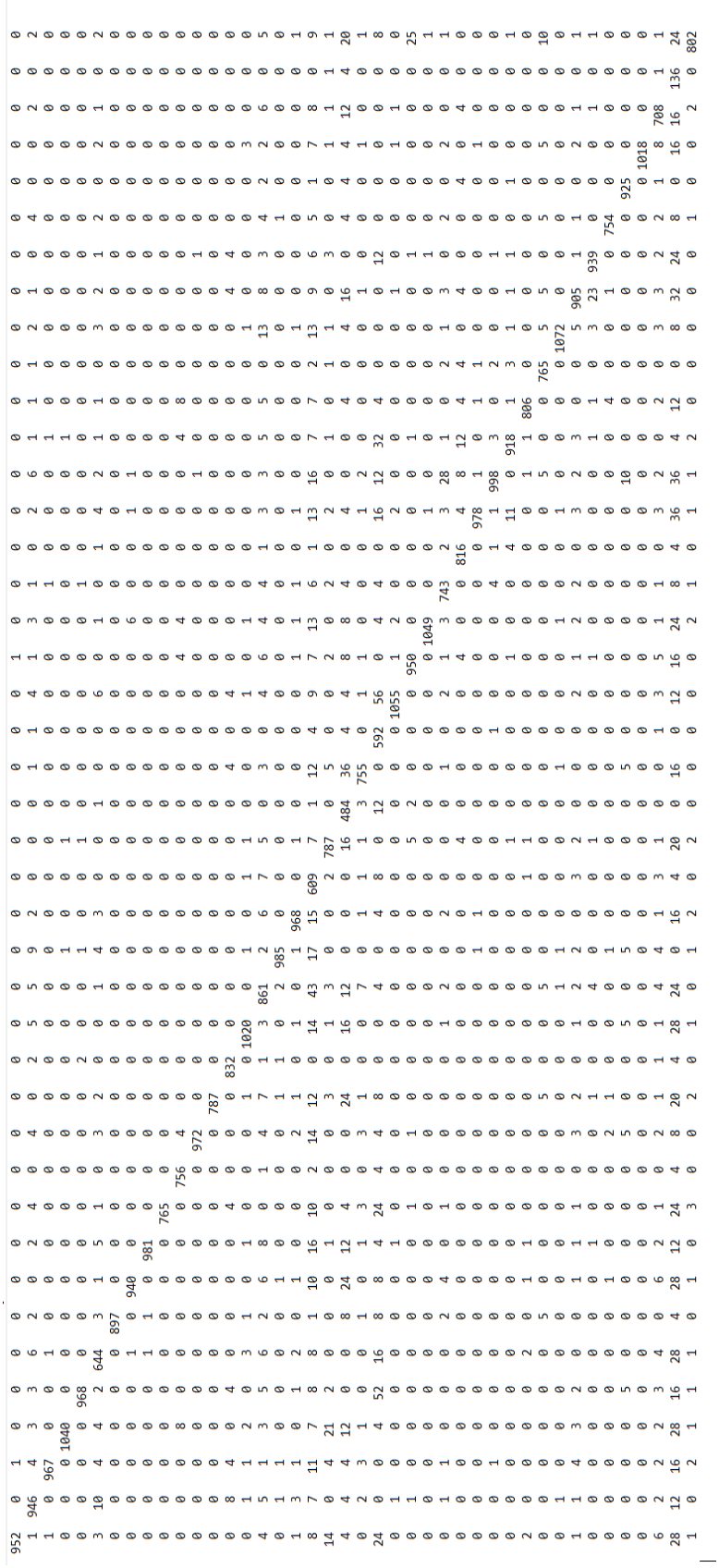


Figura 0-6 Matriz Confusión CNN 3D + Método de votación

Referencias

[Alexnet_im] Red Neuronal AlexNet, https://medium.com/ai-research-lab-kampala/alexnet-a-brief-review-14979ce7cc84 [último acceso: 7 de junio de 2020]

[Brunswick19] University of New Brunswick, "VPN-nonVPN dataset", https://www.unb.ca/cic/datasets/vpn.html [último acceso: 18 de noviembre de 2019]

[Boutaba18] R. Boutaba, Mohammad Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano y Oscar Caicedo Rendon, "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities", Journal of Internet Services and Applications, June 2018.

[Bernaille06] Laurent Bernaille, Renata Teixeira, Ismael Akodkenou, Augustin Soule, and Kave Salamatian. 2006. "Traffic classification on the fly", SIGCOMM Comput. Commun. Rev. 36, 2, pp. 23–26, April 2006.

[Drapper16] Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), pages 407-414, Rome, Italy, 2016.

[DPI_im] https://smex.org/security-heaven-privacy-hell-lebanese-telcos-introduce-deep-packet-inspection-dpi/ [último acceso: 7 de junio de 2020]

[Dropout_im] https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5 [último acceso: 7 de junio de 2020]

[Formato_ip] http://personales.upv.es/rmartin/tcpip/cap02s03.html [último acceso: 8 de junio de 2020]

[ExcelM] https://support.office.com/es-es/article/inicio-r%C3%A1pido-crear-una-macro-741130ca-080d-49f5-9471-1e5fb3d581a8 [último acceso: 10 de junio de 2020]

[Fontugne18] Romain Fontugne, Toshio Hirotsu, Kensuke Fukuda, "An image processing approach to traffic anomaly detection", AINTEC '08, Asian Internet Engineering Conference, Pattaya, Thailand, Nov 2018.

[Fernandes09] S. Fernandes, R. Antonello, T. Lacerda, A. Santos, D. Sadok and T. Westholm, "Slimming Down Deep Packet Inspection Systems," IEEE INFOCOM Workshops 2009, Rio de Janeiro, pp. 1-6, 2009.

[ISCX16] "ISCX-VPN-NonVPN-2016", https://iscxdownloads.cs.unb.ca/iscxdownloads/ISCX-VPN-NonVPN-2016/ [último acceso: 18 de noviembre de 2019]

[JZHANG19] J. Zhang, F. Li, H. Wu and F. Ye, "Autonomous Model Update Scheme for Deep Learning Based Network Traffic Classifiers," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, pp. 1-6, 2019.

[JinlinWang17] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, Zhongzhen Yang, "End-to-end Encrypted Traffic Classification with One-dimensional Convolution Neural Networks", IEEE, 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), 2017.

[KerasB] https://keras.io/ [último acceso: 3 de junio de 2020]

[KDD] http://fcojlanda.me/es/ciencia-de-los-datos/kdd-y-mineria-de-datos-espanol/ [último acceso: 10 de junio de 2020]

[Lopez-Martin17] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, Jaime Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things", IEEE Access. vol. 5, pp. 18042-18050, September 2017.

[Lotfollahi12] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, Mohammdsadegh Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning", Soft Comput 24, 1999–2012 (2020).

[Larriva20] Xavier Larriva-Novo, Mario Vega-Barbas, Victor Villagra, Diego Rivera, Manuel Álvarez-Campana, and Julio Berrocal, "Efficient Distributed Preprocessing Model for Machine Learning-Based Anomaly Detection over Large-Scale Cybersecurity Datasets", Applied Sciences. Vol 10, pp. 3430, July 2020.

[LSTM_im] https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/ [último acceso: 11 de junio de 2020]

[Malhotra15] P Malhotra, L Vig, G Shroff, P Agarwal, "Long Short Term Memory Networks for Anomaly Detection in Time Series", ESANN, 2015.

[PanWang19] Pan Wang, Xuejiao Chen, Feng Ye and Zhixin Sun, "A Survey of Techniques for Mobile Service Encrypted Traffic Classification Using Deep Learning", IEEE Access, vol. 7, pp. 54024-54033, April 2019.

[Possebon19] I. P. Possebon, A. S. Silva, L. Z. Granville, A. Schaeffer-Filho and A. Marnerides, "Improved Network Traffic Classification Using Ensemble

Learning," 2019 IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, pp. 1-6, 2019.

[Puerto] https://interpolados.wordpress.com/2017/03/27/numeros-de-puerto/ [último acceso: 11 de junio de 2020]

[Quin_es18] Juan Luis García García, "Concurrencia de flujos según el mix de aplicaciones de red", TFM, Universidad Autónoma de Madrid, septiembre, 2018

[RFC_IP80] J. Postel, "Internet Protocol", RFC 760, IETF, January 1980.

[RFC_UDP80] J. Postel, "User Datagram Protocol", RFC 768, IETF, January 1980.

[RFC_MAC80] S. Sivabalan, S. Boutros, Cisco Systems, Inc., H. Shah, Ciena Corp., S. Aldrin, Google Inc., M. Venkatesan, Comcas"Media Access Control", RFC 7769, IETF, February 2016.

[RFC_TCP80] J. Postel, "Transmission Control Protocol", RFC 761, IETF January 1980.

[ShahbazR19] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," in IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019.

[Shahbaz19] Shahbaz Rezaei and Xin Liu, "Multitask Learning for Network Traffic Classification", CoRR, June 2019.

[Spyder] https://www.spyder-ide.org/ [último acceso: 11 de junio de 2020]

[Tasa_aprend] https://elvex.ugr.es/decsai/deep-learning/slides/NN4%20Training.pdf [último acceso: 11 de junio de 2020]

[Tensor] https://www.tensorflow.org/ [último acceso: 11 de junio de 2020]

[TVT] https://es.quora.com/Qu%C3%A9-es-el-escenario-de-entrenamiento-validaci%C3%B3n-y-prueba-de-conjuntos-de-datos-en-aprendizaje-autom%C3%A1tico [último acceso: 11 de junio de 2020]

[Usama19] M. Usama, J. Qadir, A. Raza, H. Arif, K. A. Yau, Y. Elkhatib, A. Hussain, Al-Fuqaha., "Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges", IEEE Access, vol. 7, pp. 65579-65615, 2019.

[WeiWang17] Wei Wang, Ming Zhu, "Malware Traffic Classification Using Convolutional Neural Network for Representation Learning", IEEE, 2017 International Conference on Information Networking (ICOIN), 2017.

[WeinaNiu1919] Weina Niu, Zhongliu Zhuo, Xiaosong Zhang, Xiaojiang Du, Guowu Yang, Mohsen Guizani, "A Heuristic Statistical Testing Based Approach for Encrypted Network Traffic Identification", IEEE Transactions on Vehicular Technology vol. 68, no. 4, pp. 3843-3853, April 2019.

[Wire] https://www.wireshark.org/ [último acceso: 8 de junio de 2020]

Abreviaturas y acrónimos

ADDR *Address*, Dirección

AI *Artificial Intelligent*, Inteligencia artificial

API *Application Programming Interface*, Interfaz de programación de aplicaciones

AIM *AOL Instant Messenger*, Mensajería instantánea AOL

CNN *Convolutional Neural Network*, Red neuronal convolucional

DST *Destination*, Destino

DPI *Deep Packet Innspection*, Inspección profunda de paquetes

FN *False Negative*, Falso negativo

FP *False Positive*, Falso positive

FTP *File Transfer Protocol*, Protocolo de transferencia de archivos

GCC *GNU Compiler Collection*, Colección de compiladores GNU

IP *Internal Protocol*, Protocolo de internet

IMAP *Internet Message Acces Protocol*, Protocolo de acceso a mensajes de internet

KDD *Knowledge Discovery in Databases*, Extracción de conocimientos

LSTM *Long short-term memory*, Memoria a corto y largo plazo

MATLAB *Matrix Laboratory*, Laboratorio de matrices

P2P *Peer to Peer*, Red de pares

RNN *Recurrent Neural Network*, Red neuronal recurrente

RAM *Random Access Memory*, Memoria de acceso aleatorio

RGB *Red, Green and Blue*, Rojo, verde y azul

SRC *Source*, Fuente

SFTP *SSH File Transfer Protocol*, Protocolo de transferencia de archivos SSH

SMPTS *Simple Mail Transfer Protocol*, Protocolo para transferencia simple de correo

TFM Trabajo de Fin de Máster

TCP Transmission Control Potocol, Protocolo de control de transmisión

TP *True Positive*, Verdadero positivo

TN *True Negative*, Falso positivo

UDP	*User Datagram Protocol, Protocolo de datagrama de usuario*
URL	*Uniform Resource Locator,* Localizador de recursos uniforme
VPN	*Virtual Private Network,* Red privada virtual